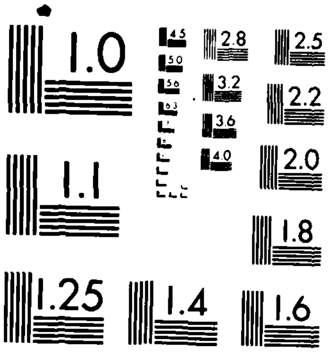
AD-A162 305 ANALYSIS OF TWO ADVANCED SMOOTHING ALGORITHMS(U) NAVAL POSTGRADUATE CCHOOL MONTEREY CA J A VASQUEZ SEP 85 1/2

UNCLASSIFIED F/G 12/1 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ANALYSIS OF TWO ADVANCED
SMOOTHING ALGORITHMS

by

Jose A. Vasquez, Jr.

September 1985

Thesis Advisor: Peter A. W. Lewis

Approved for public release; distribution is unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (*When Data Entered*)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.
AD-A162 305

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (*and Subtitle*)
Analysis of Two Advanced Smoothing Algorithms

5. TYPE OF REPORT & PERIOD COVERED
Master's Thesis
September, 1985

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
Jose A. Vasquez, Jr.

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943-5100

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93943-5100

12. REPORT DATE
September, 1985

13. NUMBER OF PAGES
165

14. MONITORING AGENCY NAME & ADDRESS(*If different from Controlling Office*)

15. SECURITY CLASS. (*of this report*)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (*of this Report*)

Approved for public release; distribution is unlimited.

17. DISTRIBUTION STATEMENT (*of the abstract entered in Block 20, if different from Report*)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (*Continue on reverse side if necessary and identify by block number*)

Neighborhood, Window Size, Span Value, Cross-Validation

20. ABSTRACT (*Continue on reverse side if necessary and identify by block number*)

This thesis examines two smoothing algorithms which deviate from the classical method of using only one neighborhood size in the smoothing procedure. The Supersmooth algorithm uses three neighborhood sizes with local cross-validation in order to estimate an optimal neighborhood size. The Split Linear Fit algorithm uses any number of neighborhood sizes and computes a family of linear fits corresponding to each neighborhood size; the final smooth points are a weighted average of the linear fits. These two

20. advanced smoothers are evaluated against the results produced by previously validated, commonly used smoothers and regression techniques. The measure of performance is the quality of the smooth curves and the value of the sum of squared residuals.

Approved for public release; distribution is unlimited.

Analysis of Two Advanced Smoothing Algorithms

by

Jose A. Vasquez, Jr.
Captain, United States Army
B.S., Texas A&I University, 1975

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
September 1985

Author: ______________________________
Jose A. Vasquez, Jr.

Approved by: ______________________________
Peter A. W. Lewis, Thesis Advisor

____
Paul S. Fischbeck, Second Reader

____
A. R. Washburn, Chairman,
Department of Operations Research

K.T. Marshall
____
Kneale T. Marshall,
Dean of Information and Policy Sciences

ABSTRACT

This thesis examines two smoothing algorithms which deviate from the classical method of using only one neighborhood size in the smoothing procedure. The Supersmooth algorithm uses three neighborhood sizes with local cross-validation in order to estimate an optimal neighborhood size. The Split Linear Fit algorithm uses any number of neighborhood sizes and computes a family of linear fits corresponding to each neighborhood size; the final smooth points are a weighted average of the linear fits. These two advanced smoothers are evaluated against the results produced by previously validated, commonly used smoothers and regression techniques. The measure of performance is the quality of the smooth curves and the value of the sum of squared residuals.

[illegible]

THESIS DISCLAIMER

The reader is cautioned that computer programs developed in this research may not have been exercised for all cases of interest. While every effort has been made, within the time available, to ensure that the programs are free of computational and logic errors, they cannot be considered validated. Any application of these programs without additional verification is at the risk of the user.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to take this opportunity to thank a few of the people who have helped me make this thesis possible. First I say "thank you" to Dr. P.A.W. Lewis who truly has guided me in order to produce a quality, informative thesis. Also, "thank you" LCDR Paul S. Fischbeck for your generous, informative comments as the second reader. Next I would like to say that we are indebted to Dr. P.D. Welch and Dr. P. Heidelberger for having the use of GRAFSTAT on a test bed basis at the Naval Postgraduate School. I would like to thank the many helpful, friendly, and courteous people at W.R. Church Computer Center, like Dennis Mar, Larry Frazier, and the computer operators who are too numerous to name individually. Lastly, I say 'Muchas Gracias, Mi Amores' to my loving wife and son who have been very patient with me over the last two years. And a 'Thank you' to those other persons that I may have forgotten.

I. INTRODUCTION

A. BACKGROUND

"It is a well-established rule of scientific investigation that the first time an experiment is performed the results bear all too little resemblance to the 'truth' being sought" [Ref. 1: p. 1]. The experiment may be the simple task of data collection, i.e. survey, or a process of generating data. The analyst may have a small set or a large set of data or a series of observations which must be analyzed. After some data analysis, the analyst may extract quantities relevant to purposes that he/she has in mind for further analysis. This analysis and data extraction process has the formal name of data reduction. Tukey calls this process "exploratory data analysis" [Ref. 2: p. 1].

There are several statistical methods that can facilitate the data reduction process. The quote, "a picture says a thousand words," suggests that the data analysis involves pictorial representations of the observed data. The single, most powerful statistical tool is a "well-chosen graph" [Ref. 3: p. 1]. A well-chosen graph enables salient features of a data set to be picked out and vividly portrayed so that the analyst can spot the features of particular interest [Ref. 4: p. 41].

The data set is very often bivariate data, i.e. pairs of values $(X_1, Y_1), \ldots, (X_N, Y_N)$, where it is conventional that the Y_I, called the ordinate, be a function of the corresponding X_I, called the abscissa. The abscissa indicates a specific snapshot of time or is the input to an experiment, i.e. the value of an independent variable. The analysis of the data basically concentrates on finding a relationship between the X_I and the Y_I. The single most powerful statistical tool for analyzing the relationship between the X_I and the Y_I is the scatterplot [Ref. 3: p. 75]. A scatterplot is a two-dimensional graph which visually displays the relationship of the pairs of X_I and Y_I. The vertical axis of the scatterplot represents the scale values of the ordinate or Y_I, and the horizontal axis represents the scale values of the abscissa or X_I. A scatterplot is easily accepted by the human brain which quickly summarizes the depicted information and extracts the salient features, patterns, and relationships that are not detected with other data analytical methods, e.g. tabulated data. Figure 1.1 is an example of a scatterplot displaying the Daily Sea-Surface Temperature for 1971 at Granite Canyon, just South of Point Sur, California [Ref. 4]. This sea-surface temperature data for 1971 is given in tabular form in Appendix D.

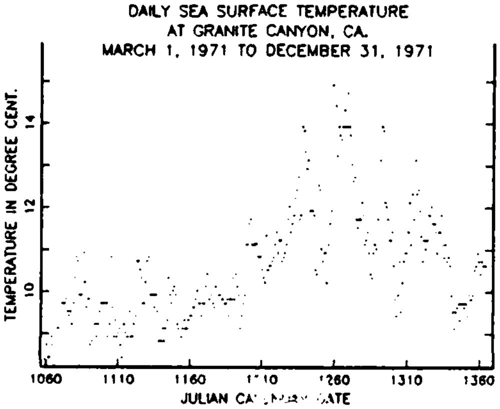


Figure 1.1 Scatterplot of Sea Surface Temperatures for 1971.

The scatterplot shown in Figure 1.1 is more compact and informative than the corresponding tabulated data in Appendix D. The scatterplot indicates that the sea-surface temperature varies with the time of year, i.e. general temperatures increase during the summer and decrease during the fall. There may have been other extraneous factors that affected the temperatures, e.g. the warm ocean current El Nino, an intra-yearly occurrence which sometimes causes great climatic turbulence all over the world, could be the cause of the great temperature variations shown by the scatterplot in Figure 1.1. It is very difficult to

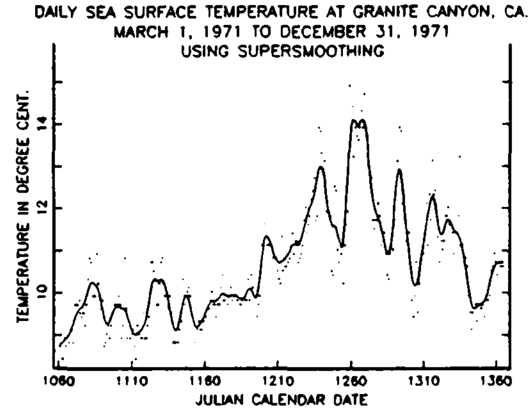


Figure 1.2 Smoothed Sea Surface Temperatures for 1971.

interpret the function imbeded in the scatterplot of Figure 1.1. An attempt to sketch a rough line that follows the curvature of the points may result in a tenuous and perhaps incorrect line in terms of depicting the variability such as the periodicity in the data set. The sketching of the line through the scatterplot of Figure 1.1 would take time and involve strong subjective decisions. The result could be a misinterpretation of the scatterplot/data. A more effective and substantiated method of data reduction is "smoothing", see Figure 1.2. This scatterplot with a smooth curve through the raw data is more acceptable to the human eye than a plain, data scatterplot and fairly well approximates the raw data. A cyclic change of the sea surface temperature is emphasized by the smooth curve. In addition, Figure 1.2 depicts that a cycle with roughly a monthly period could exist in the data, i.e. there are twelve peaks shown by the smooth curve.

Smoothing can be used on data sets whose scatterplots indicate an underlying relationship that is either a simple linear function or a complex sinusoid function. Smoothing has in the recent past years become a useful data reduction technique. Banks, insurance companies, and industrial firms smooth economic surveys [Ref. 5: p. 1].

The government does smoothing on income data such as tax payments, salaries and benefits to civil servants, and social services costs [Ref. 6]. The space program does smoothing of test flight paths, fuel usage, and orbiting ejection path data [Ref. 7]. Conferences have had smoothing as the sole subject of discussion [Ref. 8].

The smoothing algorithm that is used to smooth a data set must use a procedure that is flexible enough to discover trends in the data, i.e. be able to accurately trace the observed data and respond to local changes. Therefore, the algorithm should use local smoothing rather than global smoothing which is used in linear regression and curve fitting. This procedure allows the observed data to determine the shape of the smooth curve.

An advanced smoothing algorithm must be more computationally efficient and more user friendly than most current smoothing algorithms. In addition an advanced smoothing algorithm must be able to correctly extract the underlying function from the observed data.

B. SCOPE

This paper discusses and analyzes two advanced smoothing algorithms, the Supersmoother algorithm [Ref. 9] and the Split Linear Fit Algorithm [Ref. 10]. These two smoothing algorithm were developed at Stanford University and thus have many similarities. The basic concept used in these algorithms is that the underlying function is thought of as a low frequency signal; therefore, the observed raw data is the signal plus noise. Thus, the smoother is analogous to a low-pass filter which is designed to compromise between the signal extracted, i.e. desirable effects, and the noise filtered out, i.e. undesirable effects [Ref. 10: p. 1]. Equation 1.1 shown below is a generalization of the low-pass filter:

$$Y_I = f\,(Y_I) + r_I \;, \qquad (1.1)$$

where Y_I is the observed value, $f(Y_I)$ is the smooth function or extracted signal, and r_I is the additive residual or noise filtered out. It is initially assumed that the set of Y_I is an independent and identically distributed (i.i.d.) random sample from some unknown joint distribution F(X, Y). It is also sometimes assumed that the r_I are i.i.d. with zero expectation and constant variance σ^2, but possibly correlated; thus the notation follows the convention set by time series theory [Ref. 3: p 246]. The computed smooth values are estimates of the smooth function $f(Y_I)$. It is best that the smooth point values be computed using local averaging [Ref. 10: p. 3], in other words the I^{th} smooth point value is the average of the Y values corresponding to the X values within a neighborhood of size K about X_I, where a neighborhood of size K about X_I will have (K/2) point values to the right and left. Equation 1.2 shown below states this averaging procedure in conditional form, indicating that only the Y values that correspond to the X values within the neighborhood K about X_I are involved in the averaging.

$$s(X_I) = \text{average}(Y_J \text{ given } X_J \text{ a member of the neighborhood } K_I), \qquad (1.2)$$

where $s(X_I)$ is the computed smooth point value corresponding to X_I, K_I is the neighborhood size corresponding to X_I, J=1, . . ., K_I is the J^{th} member of the neighborhood of size K_I, and I=1, . . ., N is the index of the N points to be smoothed. For the simple, ***equal-weight, moving average*** smoother, the smooth value at point X_I is computed by equation 1.3:

$$s(X_I) = \frac{1}{K} \times \sum_{J=I-\frac{K}{2}}^{I+\frac{K}{2}} Y_J \ , \qquad (1.3)$$

where K is the neighborhood size and may encompass a fraction of the data set to be smoothed or the entire data set. By looking at equation 1.3, it can be deduced that when I=1, . . ., (K-1) and I=(N-K-1), . . ., N the subscript of Y is negative and has no corresponding Y values.

Most simple moving average smoothers do not involve the latter mentioned index values and begin the averaging with I=(K/2) and end the averaging with I=N-(K/2); thus, the smooth output will have less values than N, exactly K less values.

The neighborhood size, denoted above by K, referred to later in this thesis as bandwidth, span, or windowsize, is a critical value which must be chosen carefully because it determines to a great degree, the goodness of fit of the smooth curve to the raw data. For example, with the equal-weight, moving average smoother, a large neighborhood size results in the loss of many smooth point values, and thus, the raw data is not well depicted. A commonly used measure of goodness of fit is the sum of squared residuals; thus, it is necessary to examine a squared residual value in general terms. If the output of the smooth function $f(X_I)$ is accepted as an estimate of the corresponding Y_I and a linear fit is done on the points within the neighborhood, then the expected squared residual at point X_I, given a neighborhood size K, may be determined by equation 1.4:

$$r^2(X_I \mid K) = \left[f\,(Y_I) - \frac{1}{K} \times \sum_{J=I-\frac{K}{2}}^{I+\frac{K}{2}} f\,(Y_J) \right]^2 + \frac{1}{K} \times \sigma^2 \ . \qquad (1.4)$$

The term within the brackets is the bias component of the estimated residual value corresponding to X_I; in other words, the degree to which the smooth point value deviates from the actual point value. The second term is the variance component which indicates that the assumed inherent constant variance of the residuals must be equally shared by the estimated residuals within the neighborhood. Increasing the neighborhood size, K, increases the bias and decreases the variance, thus a plot of the smooth values will get smoother as K is increased. Decreasing K will have the opposite effect.

Most smoothing algorithms use only one neighborhood size to produce the smooth values, i.e. the same K for all X_I in equations 1.3

and 1.4. The problem with this method is that the smoothing program may have to be run several times with each run containing a different K before the desired smoothing effect is produced. This thesis discusses two advanced smoothing algorithms which deviate from this procedure. The two advanced smoothing algorithms are:

1. the Supersmoother algorithm developed by Friedman and Stutzle [Ref. 9];
2. the Split Linear Fit algorithm developed by McDonald and Owen [Ref. 10].

The Supersmoother requires that the user enter three different neighborhood sizes, $SPAN_1$, $SPAN_2$, and $SPAN_3$, in increasing order. Each span value determines a neighborhood size about each X_I on which a linear regression is done. Therefore, three sets of regression results will correspond to each X_I. Each of the three slope values, the three corresponding y-intercept value, and the corresponding X_I are used to compute three fitted values. Each fitted value is subtracted from the input Y_I value corresponding to X_I; the resulting values are called cross-validated residuals. The minimum, absolute value of these cross-validated residuals is then selected along with its span value. This span value is an estimate of the optimal span value corresponding to X_I. This estimate is then adjusted using an outlier rejection rule which will reflect the degree of robust smoothing desired by the user. The smallest span value, $SPAN_1$, and the largest span value, $SPAN_3$, dictate the range within which Supersmoother finds the optimal span value. The middle span value, $SPAN_2$, is used as a central smoother, i.e. by smoothing the array of optimal span values with the middle span value the variability is reduced. This smoothing adjusts the span values so that the values flow smoothly from one point to the next adjacent point. This method of finding the optimal span values is called local cross-validation [Ref. 9: p. 1]. The method of cross-validation is a testing procedure that uses the estimated regression equation on data different than the data used to estimate the coefficients of the estimated regression equation [Ref. 11: p. 110].

The Split Linear Fit algorithm uses one or more neighborhoods, called window sizes, in order to produce for each I a family of linear fitted values. Weights which indicate the goodness of fit of the fitted values are then assigned to each of these linear fitted values. The final I^{th} smooth value is computed as a weighted average of the linear fits within the I^{th} family of linear fits.

This technique of using more than one neighborhood size allows the analyst to set upper and lower limits on the neighborhood size. By accepting more than one neighborhood size, these advanced smoothers take full advantage of the powerful computational capabilities of a computer and thus are quicker and more efficient than other smoothers, i.e. desired smoothing effects are achieved in less runs of a smoothing program.

The purpose of this thesis is to expand the data smoothing subroutine developed by Friedman and Stuetzle [Ref. 9] and the smoothing program developed by McDonald and Owen [Ref. 10] into user friendly, interactive computer programs, i.e. the user exchanges information with the computer, that can be used as an exploratory data analytical tool by students and faculty of the Naval Postgraduate School.

The Supersmoother algorithm was written as a FORTRAN subroutine and has been incorporated into an interactive FORTRAN program. The Split Linear Fit algorithm was part of a data smoothing package written in the C computer language, which is not a common computer language used at the Naval Postgraduate School. The Split Linear Fit algorithm has been translated and is incorporated into an interactive FORTRAN program. The point values produced by the Split Linear Fit FORTRAN version are equivalent to the point values produced by the C language version. Both the Supersmoother and the Split Linear Fit algorithms are written in FORTRAN 77 for use on the IBM 3033 computer being used at the Naval Postgraduate School. SUPSMO is the Supersmoother program and SPLITSMO is the Split Linear Fit program. These two FORTRAN programs are designed to produce cutput in any one of following three forms:

1. a CMS data file;
2. an APL, "A Programming Language," variable;
3. graphs produced with the IBM GRAFSTAT[1] statistical graphics package [Ref. 12].

These programs are written for use by any individual who has access to the IBM 3033. With simple commands the user can create or access an APL workspace and create an APL variable that stores the smooth output. Access to the GRAFSTAT graphics package is easy and done without exiting the smoothing program. Creation of a CMS file is even easier. GRAFSTAT is a graphics package which is an experimental program available at the Naval Postgraduate [Ref. 12].

Complete user instructions on how to use SUPSMO and SPLITSMO are available in Chapter VI and VII. Mathematical details on the the Supersmoother and the Split Linear Fit are presented in Chapters II and III, respectively. In Chapter IV are the evaluation results from smoothing three simple sets of data with these two advanced smoothers. These smoothing results are compared to the smoothing produced by previously verified smoothers, e.g. LOWESS and Moving Average. In Chapter V a real application of the Supersmoother and the Split Linear Fit programs is presented. The Granite Canyon Daily Sea-Surface Temperature data for the period of March 1971 to February 1983 is used in the analysis presented in Chapter V. This data set is used because of the large size of the series, 4380 points; because the variance may not be constant, and because the complex underlying function seems to contain some periodicity.

[1] GRAFSTAT is an experimental APL package from IBM which the Naval Postgraduate School is using under an agreement with the IBM Research Center, Yorkstown Heights, N. Y.

II. TECHNICAL DESCRIPTION OF SUPERSMOOTHER ALGORITHM

A. OVERVIEW

The data smoothing algorithm, Supersmoother, was developed at Stanford University by Jerome H. Friedman and Werner Stuetzle [Ref. 9]. The smoothing technique uses local averaging [Ref. 9: p. 3], local linear fitting [Ref. 9: p. 3], and selection of a local optimal span [Ref. 9: p. 8], i.e. application of method of cross-validation [Ref. 9: p. 1]. The developers claim that Supersmoother is "both very flexible and rapidly computable" [Ref. 9: p. 3]. One of the features which makes Supersmoother flexible is that Supersmoother is scale independent. In other words, the X values must be equi-spaced but can belong to the interval (0.0, 1.0] or the interval [1.0, 2.0, 3.0, . . ., N], where N is the number of point values to be smoothed, while the Y values must be real values and need not be equi-spaced. Another feature which makes Supersmoother flexible is that there is an option of entering one or three global span values where these values are entered as a ratio of the span to the number of points to be smoothed. Another flexibility feature is that there is an outlier rejection rule which allows the user to adjust the degree of robustness using an index within the interval [0.0, 10.0], where 0.0 indicates robust smoothing and 10.0 indicates non-robust smoothing. Supersmoother uses a small amount of computer time and of storage space by using computation and data storage procedures commonly used in dynamic programming, i.e. $F_{I-1}(X)$ is used to update $F_I(X)$ and only the new value is stored.

The objective of Supersmoother is to efficiently smooth a scatterplot [Ref. 9: p. 1]. Supersmoother consists of two subroutines, the Combining Subroutine and the Smoothing Subroutine, see Figure 2.1. The Combining Subroutine and the Smoothing Subroutine exchange data arrays once if only one span value is used and eight times if three span values are used.

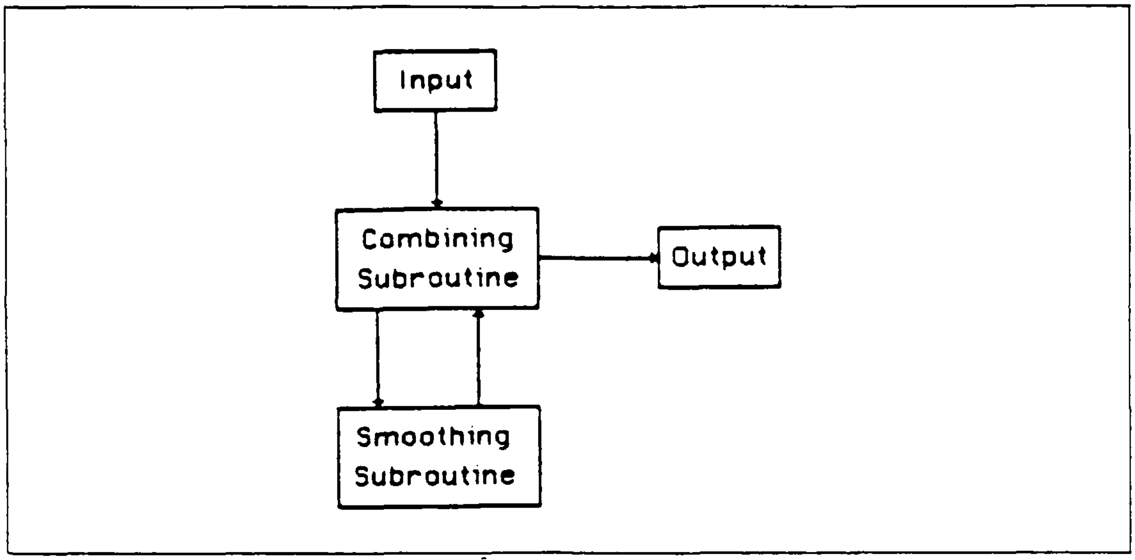


Figure 2.1 Supersmoother Subroutines.

Figure 2.2 shows the flow of data within the Supersmoother algorithm. The Combining Subroutine receives the data to be smoothed and other pertinent parameters and sets up the data for transmission to the Smoothing Subroutine. The Smoothing Subroutine smoothes the data array three times, using each span value once, and then computes the residual values corresponding to the three smoothed arrays. Then each array of residual values is smoothed using SPAN_2 in order to reduce the total variability and create smooth transitions between adjacent residual values. The smoothed residual values are then returned to the Combining Subroutine where the optimal span values are determined and adjusted using the outlier rejection rule. The adjusted optimal span values are then sent back to the Smoothing Subroutine for smoothing with SPAN_2. This is done so that variability between the values will again be reduced. The now smoothed, adjusted, optimal span values are returned to the Combining Subroutine where they are used in an interpolation procedure. The results of this interpolation procedure are estimates of the final smoothed values. These estimated smoothed values are then returned to the Smoothing Subroutine for smoothing with SPAN_1 in order to reduce the variability of these values. This procedure will also accentuate outliers in the raw data because of the small neighborhood size of $SPAN_1$. The final smoothed values are returned to the Combining Subroutine which forwards the results to the user's main program for his/her use.

The Combining Subroutine does the following:

1. keeps track of pertinent computational results;
2. computes the interquartile range of the abscissa;
3. defines zero for computational and comparative purposes;
4. determines the optimal span corresponding to each abscissa;
5. applies the outlier rejection rule in order to adjust the robustness;
6. estimates the smoothed output.

If only one span value is used, only the first three items of the above list are executed by the Combining Subroutine, and the Smoothing Subroutine is used only once. If three span values are used, all the items are executed.

The smoothed output produced by the Smoothing Subroutine is not the final smoothed values given to the user. Therefore, to be able to distinguish the output forwarded to the user, i.e. the smoothed Y values, from the smoothed values exchanged between the subroutines, any array to be smoothed by the Smoothing Subroutine, e.g. the residual values, will be called Z within the Smoothing Subroutine. Z. After the array is smoothed and returned to the Combining Subroutine, it regains it's usual name. The Smoothing Subroutine does the following:

1. computes the neighborhood size, (IT), the number of points to be included in the local averaging;
2. computes the base mean, variance, and covariance values that will be used in the computation of the smoothed values Z_I;
3. computes the smoothed values Z_I at the beginning of the data array, i.e. the first (IT/2) smoothed points that are not usually computed by most smoothers;

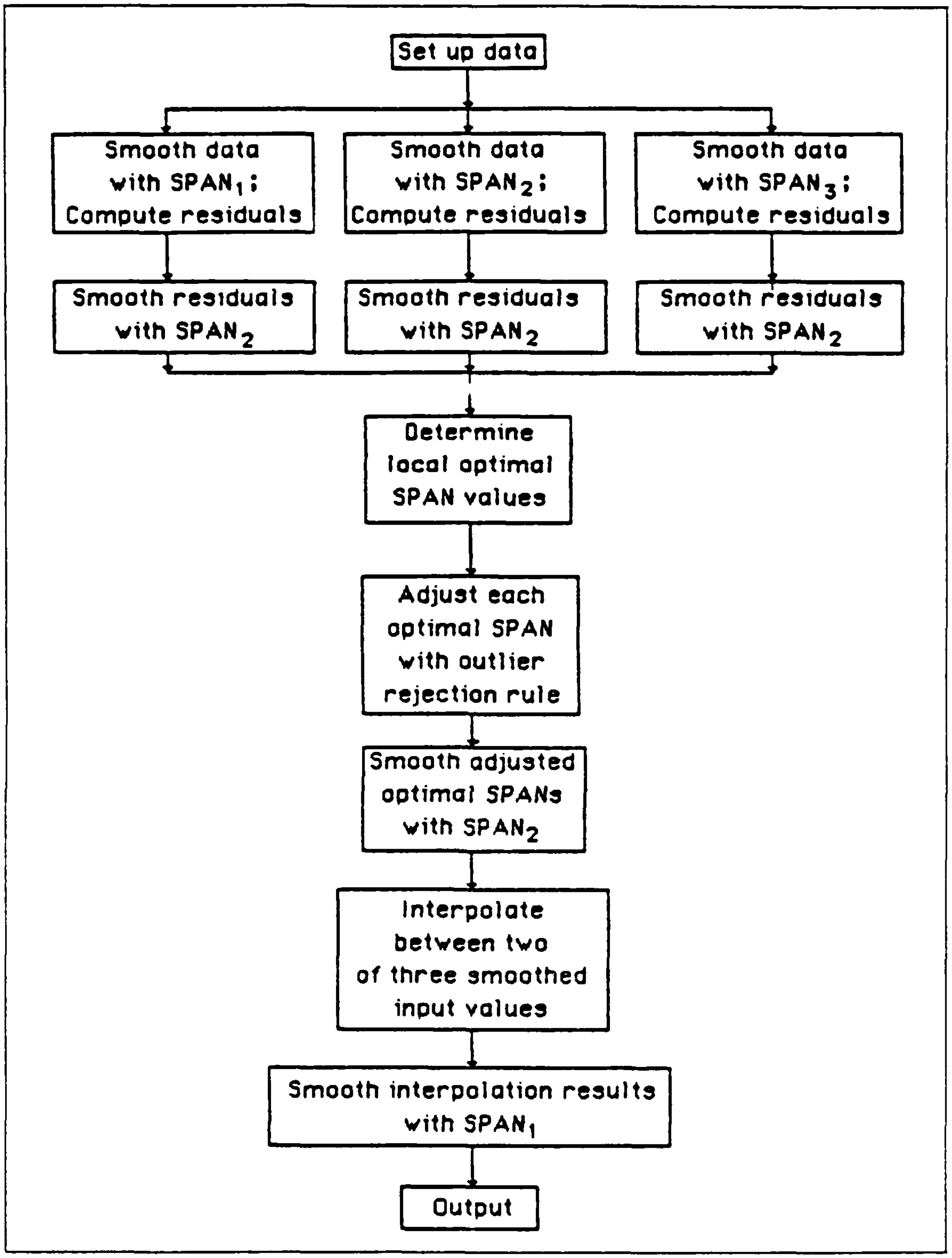

Set up data
Smooth data with SPAN₁; Compute residuals
Smooth data with SPAN₂; Compute residuals
Smooth data with SPAN₃; Compute residuals
Smooth residuals with SPAN₂
Smooth residuals with SPAN₂
Smooth residuals with SPAN₂
Determine local optimal SPAN values
Adjust each optimal SPAN with outlier rejection rule
Smooth adjusted optimal SPANs with SPAN₂
Interpolate between two of three smoothed input values
Smooth interpolation results with SPAN₁
Output

Figure 2.2 Data Flow of Supersmoother.

4. computes the smoothed values Z_I for the middle of the data array;
5. computes the smoothed values Z_I for the end of the data set, i.e. the last (IT/2) smoothed points that are not usually computed by most smoothers.

The smoothed values Z_I are the result of a local linear fit within the neighborhood of points about X_I, the abscissa corresponding to the smoothed Z_I. The Smoothing Subroutine computes the cross-validated residuals only at the time when the input data is smoothed, see Figure 2.2.

B. MATHEMATICAL DETAILS---COMBINING SUBROUTINE

The Combining Subroutine requires the following user input:

1. N, the number of points to be smoothed;
2. $Y_1, \ldots, Y_N$, the point values that need smoothing;
3. $X_1, \ldots, X_N$, the abscissa corresponding to the Y_I values if the abscissa do not belong to the interval [1.0, 2.0, . . ., N];
4. IPER, equals 1 or 2, to indicate that the abscissa belong to the interval [1.0, 2.0, . . ., N] or the interval (0.0, 1.0], respectively;
5. the span values $SPAN_1$, $SPAN_2$, and $SPAN_3$;
6. ALPHA, the outlier rejection rule index.

The Smoothing Subroutine assumes that the input data set is in chronological order , i.e. Y_I occurred before Y_{I+1} where I=1, . . ., (N-1); thus, the abscissa will be in increasing order.

The abscissa interquartile range, SCALE, is computed using equations 2.1 through 2.3:

$$I = \frac{N}{4} ; \tag{2.1}$$

$$J = 3 \times I ; \tag{2.2}$$

$$\mathrm{SCALE} = X_J - X_{I'} \ . \tag{2.3}$$

If N<4 and if the computer allows indices with value of zero, then SCALE=0.0, otherwise an error is created. In order to define zero, VSMLSQ, SCALE must be greater than zero. If SCALE≤0.0, then J=J+1 and SCALE is recomputed using equation 2.3. Zero is defined by equation 2.4:

$$\mathrm{VSMLSQ} = [(1\times10^{-3})\times\mathrm{SCALE}]^2 \ . \tag{2.4}$$

If only one span value is used, the Smoothing Subroutine is called by the Combining Subroutine, and the smoothed data array returned is the smoothed Y_I, see Figure 2.1. If this procedure is used, the smoothed Y_I will have too much variability [Ref. 9: p. 9] and will be very robust, so it is best to use three span values.

When three span values are used, the input Y_I are smoothed three times, once with each span value, see Figure 2.2. For ease of discussion, Y_{IS} will be used to indicate the I^{th} input Y value smoothed using $SPAN_S$, where I=1, . . ., N and S=1, 2, 3. As mentioned before, during the smoothing of the input Y_I, cross-validated residual values are computed. These residual values will be identified by $ACVR_{IS}$, i.e. the I^{th} cross-validated residual computed when $SPAN_S$ was used. In order to reduce the variability of the smoothed Y_I, the $ACVR_{IS}$ are smoothed using $SPAN_2$, see Figure 2.2. For stability reasons an array containing the absolute value of the $ACVR_{IS}$ is smoothed. [Ref. 9: p. 9]. After the smoothing of the absolute value of the $ACVR_{IS}$, each abscissa X_I, has the following seven corresponding values:

1. the input Y_I;
2. Y_{I1} and $ACVR_{I1}$;
3. Y_{I2} and $ACVR_{I2}$;
4. Y_{I3} and $ACVR_{I3}$.

Next follows the basis of the local cross-validation method. First for each I the minimum of $ACVR_{I1}$, $ACVR_{I2}$, and $ACVR_I$ is selected and designated $ACVR_{min}$. Recalling that the second subscript of $ACVR_{IS}$ indicates the $SPAN_S$ used to smooth the input data set and produce the corresponding $ACVR_{IS}$, then the $SPAN_S$ used to produce $ACVR_{min}$ can be determined and designated SC_I. Then the outlier rejection which consists of equations 2.5 and 2.6, shown below, can be used to adjust SC_I, i.e. the span value used to compute $ACVR_{min}$, in order to reflect the degree of robustness desired by the user:

$$SC_I = SC_I + (SPAN_3 - SC_I) \times AM^{10.0-ALPHA} \; ; \qquad (2.5)$$

where

$$AM = ABS\left[\max\left(1.0\times10^{-7}, \frac{ACVR_{min}}{ACVR_{IS}}\right)\right] \; . \qquad (2.6)$$

The resulting SC_I is called the "estimated optimal span" [Ref. 9: p. 10] corresponding to I. The set of estimated optimal spans may have an unnecessarily high variance, thus they are smoothed using $SPAN_2$; the result is the set of optimal spans, SC_I.

Each SC_I value is checked using one of the two following logical statements in order to verify that the span boundaries fixed by the user are not violated:

1. if $SC_I \leq SPAN_1$, then $SC_I = SPAN_1$ or;
2. if $SC_I \geq SPAN_3$, then $SC_I = SPAN_3$.

Each SC_I value is used to estimate a smooth Y_I value by interpolating between two of the Y_{IS} values previously computed. The sign and value of F in equation 2.7 forms the basis of the interpolation.

$$F = SC_I - SPAN_2 \; . \qquad (2.7)$$

If F is negative then equations 2.8 and 2.9, shown below, are used to estimate the smooth Y_I;

$$F = \frac{-F}{SPAN_2 - SPAN_1} \; ; \qquad (2.8)$$

$$\text{estimated smooth } Y_I = [(1.0 - F) \times Y_I^2] + [F \times Y_{I1}] \ . \qquad (2.9)$$

otherwise equations 2.10 and 2.11, shown below, are used:

$$F = \frac{F}{SPAN_3 - SPAN_2} \ ; \qquad (2.10)$$

$$\text{estimated smooth } Y_I = [(1.0 - F) \times Y_I^2] + [F \times Y_{I3}] \ . \qquad (2.11)$$

The final smooth Y_I values are obtained by smoothing the estimated smooth Y_I using $SPAN_1$. This smoothing is done in order to reduce the variability of the estimated smooth Y_I caused by the variance in the input data.

C. MATHEMATICAL DETAILS---SMOOTHING SUBROUTINE, PRIMARY USE

The primary use of the Smoothing Subroutine is to smooth data with abscissa values in the interval [1.0, 2.0, . . ., N]. The secondary use of the Smoothing Subroutine is to smooth data with abscissa values in the interval (0.0, 1.0]. The Smoothing Subroutine requires that the following data be transferred from the Combining Subroutine:

1. N, the number of points to be smoothed;
2. the array to be smoothed, in this subroutine this array will be referred to as $Z_1, \ldots, Z_N$;
3. $X_1, \ldots, X_N$, the abscissa that correspond to the Z_I;
4. SPAN, the span value;
5. a flag, IPER, which indicates whether the cross-validated residuals are to be computed or not computed;
6. VSMLSQ, the defined value of zero.

The size of the neighborhood of points included in the local averaging is determined by SPAN. Most smoothers will not compute the size of the neighborhood and require that the user enter an odd integer number indicating the size of the neighborhood. Supersmoother will compute the size of the neighborhood, thus allowing the user an infinite number of choices, since the value of SPAN, as entered by the user, belongs to the interval (0.0, 1.0]. Since the computation of the neighborhood results in an integer value, different SPAN value entries may result in the same neighborhood size, e.g. if N=100, then SPAN values of 0.04 and 0.045 will both result in a neighborhood size of 4. Supersmoother uses two neighborhood sizes, IBW and IT, both integer values. The first (IBW+1) smoothed points and the last IBW smoothed points of the output Z array are computed differently than the central smoothed points. Most smoothers drop IBW points at the beginning and at the end of the smoothed Z array, e.g. the Moving Average smoother mentioned in Chapter I. IT is the number of points included in the local averaging. These IT points are the nearest neighbors of X_I, the abscissa corresponding to the smoothed point being computed. Supersmoother will always compute IT to be an odd integer. IBW, on the other hand, sometimes may be odd or even, depending on the value of N and SPAN. Since IT is odd, the X_I will be the median of the neighborhood with $\lfloor(IT/2)$, integer division, points to the left and right. The following two equations are used in the computation of the neighborhoods, IBW and IT:

$$IBW = (0.5 \times SPAN \times N) - 0.5 \ ; \qquad (2.12)$$

$$IT = (2 \cdot IBW) - 1 \ . \qquad (2.13)$$

The first IT values of the X and Z arrays are used to compute the base or initial values of X_{mean}, Z_{mean}, covariance of X and Z, and variance of X using equations 2.14 through 2.17:

$$X_{mean} = \frac{\sum_{J=1}^{IT} X_J}{IT} \ ; \qquad (2.14)$$

$$Z_{mean} = \frac{\sum_{J=1}^{IT} Z_J}{IT} \ ; \qquad (2.15)$$

$$COV_{XZ} = \sum_{J=1}^{IT}\left[(X_J - X_{mean})\times(Z_J - Z_{mean})\right] \ ; \qquad (2.16)$$

$$VAR_X = \sum_{J=1}^{IT}(X_J - X_{mean})^2 \ . \qquad (2.17)$$

The first (IBW+1) smooth Z_I are computed using the results from equations 2.14 through 2.17. The first step in the computation of these smooth Z_I is to find the slope, A, of the least squares straight line through the set of points (X_1, Z_1), . . ., (X_{IT}, Z_{IT}) [Ref. 9: p. 5]. If $VAR_X \leq VSMLSQ$, then A=0.0, otherwise equation 2.18 is used:

$$A = \frac{COV_{XZ}}{VAR_X} \ . \qquad (2.18)$$

The second step is the actual computation of the smooth Z_I using the slope, A, computed with equation 2.18, the results from equations 2.14 and 2.15 and the following linear equation:

$$\text{smooth } Z_I = A\times(X_I - X_{mean}) + Z_{mean} \ . \qquad (2.19)$$

The cross-validated residual, $ACVR_I$, are computed using the following procedure:

1. compute the portion of the neighborhood occupied by the smooth point, H, using equation 2.20:

$$H = \frac{1.0}{IT} \ ; \qquad (2.20)$$

2. if $VAR_X > VSMLSQ$, then this large degree of variability inherent in the raw data must be reflected in H using equation 2.21:

$$H = H + \frac{(X_I - X_{mean})^2}{VAR_X} \ ; \qquad (2.21)$$

3. finally the cross-validated residuals are computed with equation 2.22:

$$ACVR_I = \frac{ABS(Z_I - \text{smooth } Z_I)}{1.0 - H} \ . \qquad (2.22)$$

Recall that only the first (IBW+1) smoothed Z_I values have been computed.

In order to compute the smoothed $Z_{IBW+1}, \ldots, Z_{N-IBW}$, the neighborhood of points has to be moved from one point to the next point toward the right. This is where the dynamic programming computational procedures are useful in cutting the storage space and computer time. The results from equations 2.14 through 2.17 are updated to reflect the movement of the neighborhood, i.e. the left endpoint of the neighborhood will be dropped and the point to the right of the right endpoint will enter the neighborhood. Equations 2.23 through 2.30 are used for each I, where I=(IBW+1), . . ., (N-IBW). Equations 2.23 through 2.26 are used to drop a point from the neighborhood:

$$X_{mean} = \frac{(IT \times X_{mean}) - X_{I-IT}}{IT - 1} \; ; \qquad (2.23)$$

$$Z_{mean} = \frac{(IT \times Z_{mean}) - Z_{I-IT}}{IT - 1} \; ; \qquad (2.24)$$

$$COV_{XZ} = COV_{XZ} - \frac{IT \times (X_{I-IT} - X_{mean}) \times (Z_{I-IT} - Z_{mean})}{IT - 1} \; ; \qquad (2.25)$$

$$VAR_X = VAR_X - \frac{IT \times (X_{I-IT} - X_{mean})^2}{IT - 1} \qquad (2.26)$$

Equations 2.27 through 2.30 are used to add a point to the neighborhood:

$$X_{mean} = \frac{(IT - 1) \times X_{mean} - X_I}{IT} \; ; \qquad (2.27)$$

$$Z_{mean} = \frac{[(IT - 1) \times Z_{mean}] + Z_I}{IT} \ ; \qquad (2.28)$$

$$COV_{XZ} = COV_{XZ} + \frac{IT \times (X_I - X_{mean}) \times (Z_I - Z_{mean})}{IT - 1} \ ; \qquad (2.29)$$

$$VAR_X = VAR_X + \frac{IT \times (X_I - X_{mean})^2}{IT - 1} \ . \qquad (2.30)$$

The results from equations 2.27 through 2.30 are then used in equations 2.18 through 2.22 to compute the smoothed Z_I and the cross-validated residuals if necessary.

The X_{mean}, Z_{mean}, COV_{XZ}, and VAR_X values used to compute smooth Z_{N-IBW} are used to compute the smooth Z_I values where I=(N-IBW+1), . . ., N, i.e. the smooth values at the tail-end of the Z array. These mean and variance values are used in equations 2.18 through 2.22 in the computation of the smooth Z_I. This procedure is equivalent to the procedure used to compute the smooth Z_I, where I=1, . . ., (IBW+1).

D. MATHEMATICAL DETAILS---SMOOTHING SUBROUTINE, SECONDARY USE

The secondary use of the Smoothing Subroutine is to smooth data with abscissa values in the interval (0.0, 1.0]. The Smoothing Subroutine needs the same data and follows the same steps and equations as if the abscissa were in the interval [1.0, 2.0, . . ., N]. The exceptions are noted in this section.

When using equations 2.14 through 2.17, the first IBW points and the last (IBW+1) points of the X and Z data arrays are used to compute the initial X_{mean}, Z_{mean}, covariance of X and Z, and the variance of X values. Equations 2.14 through 2.17 are then changed in order to allow these new points to be involved in the computations. Equations 2.31 through 2.34, shown below, are the result of the change and are used in the computation of the initial values of X_{mean}, Z_{mean}, covariance of X and Z and variance of X:

$$X_{mean} = \frac{\sum_{J=N-IBW+1}^{N} X_J}{IBW + 1} + \frac{\sum_{J=1}^{IBW} X_J}{IBW} \ ; \qquad (2.31)$$

$$Z_{mean} = \frac{\sum_{J=N-IBW+1}^{N} Z_J}{IBW + 1} - \frac{\sum_{J=1}^{IBW} Z_J}{IBW} \; ; \tag{2.32}$$

$$COV_{XZ} = \sum_{J=N-IBW+1}^{N} (X_J - X_{mean}) \times (Z_J - Z_{mean}) - \sum_{J=1}^{IBW} (X_J - X_{mean}) \times (Z_J - Z_{mean}) \; ; \tag{2.33}$$

$$VAR_X = \sum_{J=N-IBW+1}^{N} (X_J - X_{mean})^2 - \sum_{J=1}^{IBW} (X_J - X_{mean})^2 \; . \tag{2.34}$$

The next step in this smoothing procedure is to drop a point from the neighborhood. In the previous Section, equations 2.23 through 2.26 were used for this task, but they cannot be used in this section because the input point counter indicates that negative index values are computed at the beginning of the computations. The negative indices are the result of the last (IBW+1) point being being used in equations 2.31 through 2.34. Thus to keep the point counter on track let K=N+I-IBW-1 and change equations 2.23 through 2.26 as indicated in equations 2.35 through 2.38, respectively. Then in order to drop a point from the neighborhood, equations 2.35 through 2.38 are used:

$$X_{mean} = \frac{(IT \times X_{mean}) - X_{K-1.0}}{IT - 1} \; ; \tag{2.35}$$

$$Z_{mean} = \frac{(IT \times Z_{mean}) - Z_{K-1.0}}{IT - 1} \; ; \tag{2.36}$$

$$COV_{XZ} = COV_{XZ} - \frac{IT \times (X_{K-1.0} - X_{mean}) \times (Z_K - Z_{mean})}{IT - 1} \; ; \tag{2.37}$$

$$VAR_X = VAR_X - \frac{IT \times (X_{K-1.0} - X_{mean})^2}{IT - 1} \; . \tag{2.38}$$

Since a point was dropped from the neighborhood the next adjacent point on the right boundary of the neighborhood must be entered into the neighborhood. In the previous Section, equations 2.27 through 2.30 performed this task, but in order to keep the point counter on track these equations must be modified. Therefore, let K=I+IBW and the new equations are equations 2.39 through 2.42. Thus to add the next point to the neighborhood, these new equations are used:

$$X_{mean} = \frac{\left[(IT - 1) \times X_{mean}\right] - X_K}{IT} \quad ; \qquad (2.39)$$

$$Z_{mean} = \frac{\left[(IT - 1) \times Z_{mean}\right] - Z_K}{IT} \quad ; \qquad (2.40)$$

$$COV_{XZ} = COV_{XZ} - \frac{IT \times (X_K - X_{mean}) \times (Z_K - Z_{mean})}{IT - 1} \quad ; \qquad (2.41)$$

$$VAR_X = VAR_X - \frac{IT \times (X_K - X_{mean})^2}{IT - 1} \quad . \qquad (2.42)$$

The results produced by equations 2.39 through 2.42 are then used in equations 2.18 through 2.22 to compute the smooth Z_I, where I=1, . . ., (IBW+1).

In order to compute the middle smooth Z_I values, i.e. smooth Z_I where I=(IBW+2), . . ., (N-IBW), equations 2.23 through 2.30 and equations 2.18 through 2.22 are used as they are, i.e. no changes involved.

The computation of the last (IBW-1) smooth Z_I values, i.e. smooth Z_I where I=(N-IBW+1), . . ., N, involves changing equations 2.23 through 2.26 a second time, in order to maintain the point counter on track. This change is needed because the first (IBW-1) input points are used to compute these smooth Z_I and the point counter must not exceed N, the number of points to be smoothed. Thus let K=I+IBW-N and the result of the change is shown in equations 2.43 through 2.46. Equations 2.44 through 2.46 are used to drop a point from the neighborhood:

$$X_{mean} = \frac{(IT \times X_{mean}) - X_K + 1.0}{IT - 1} \; ; \qquad (2.43)$$

$$Z_{mean} = \frac{(IT \times Z_{mean}) - Z_K - 1.0}{IT - 1} \; ; \qquad (2.44)$$

$$COV_{XZ} = COV_{XZ} - \frac{IT \times (X_K - 1.0 - X_{mean}) \times (Z_K - Z_{mean})}{IT - 1} \; ; \qquad (2.45)$$

$$VAR_X = VAR_X - \frac{IT \times (X_K + 1.0 - X_{mean})^2}{IT - 1} \; . \qquad (2.46)$$

In order to replace the point dropped from the neighborhood, equations 2.27 through 2.30 are used, but in a different form because of the same reason that equations 2.23 through 2.26 were changed above. Therefore, with K=I-IBW-1 the equations used to add a point to the neighborhood are equations 2.47 through 2.50:

$$X_{mean} = \frac{\left[(IT - 1) \times X_{mean}\right] + X_K}{IT} \; ; \qquad (2.47)$$

$$Z_{mean} = \frac{\left[(IT - 1) \times Z_{mean}\right] + Z_K}{IT} \; ; \qquad (2.48)$$

$$COV_{XZ} = COV_{XZ} + \frac{IT \times (X_K - X_{mean}) \times (Z_K - Z_{mean})}{IT - 1} \; ; \qquad (2.49)$$

$$VAR_X = VAR_X - \frac{IT \times (X_K - X_{mean})^2}{IT - 1} \qquad (2.50)$$

The results produced by equations 2.47 through 2.50 are then used in equations 2.18 through 2.22 in order to compute the smooth Z_I where I=(N-IBW+1), . . ., N. The disadvantage of using abscissa values from the interval (0.0, 1.0] versus abscissa from the interval [1.0, 2.0, . . ., N] is that a slight degree of distortion is produced at the ends of the smoothed Z array. The distortion could be caused by the 1.0 adjustment factor in equations 2.35, 2.36, 2.43, and 2.44.

E. SELECTION OF SPAN

The span value is the parameter that controls the smoothing of a data set. There exist no set procedures for selecting a span value. Each data analyst has his/her own method of selecting the span value. The analyst's experience with smoothers determines how the span is selected. Selection of the span value is basically a subjective process, where the analyst uses a span value which gives adequate and useful results. The user of the advanced smoothers should develop a consistent, span selection process. A common procedure used by some expert smoothers starts by looking at a scatterplot of the raw data. Then the analyst looks for periodicity and cyclic changes present in the data. This information is then used to estimate the span value to be used in the smoothing. For example, if a data set displays a cycle of about 24 points, then the span to use should be about 24/N, where N is number of points to be smoothed. This span value is a good estimate because the raw data is permitted to determine the shape of the smooth results. This procedure is used in Chapter V of this thesis in the smoothing of a large set of sea-surface temperatures.

Supersmoother is unique among smoothing algorithms in that three span values, i.e. $SPAN_1$, $SPAN_2$, and $SPAN_3$, may be entered by the user. Supersmoother will then select an optimal span value within the range of the smallest span value and the largest span value by using the method of cross-validation which was explained earlier in this Chapter. This option within Supersmoother lets the user be very flexible in the selection of the span values to use. But the user must be careful about which span values to use with Supersmoother. It is best to first try span values of 0.05, 0.2, and 0.5, as recommended by Friedman and Stuetzle [Ref. 9: p. 9]. This range of span values gives Supersmoother good coverage of the data. After viewing the results produced by Supersmoother, the user can adjust the span values in order to get the desired smooth effect. When adjusting, the user must bear in mind the bias/variance trade-off discuss earlier in this chapter. The trade-off being that if the span value is increased, then result is a smoother looking curve, while the reverse occurs when decreasing the span value.

No matter what rule is followed to determine the span values used in Supersmoother, the final smooth results accepted are based on subjective needs, applications, and preferences.

III. TECHNICAL DESCRIPTION OF SPLIT LINEAR FIT ALGORITHM

A. OVERVIEW

The Split Linear Fit smoothing algorithm was developed at Stanford University by John A. McDonald and Art B. Owen. [Ref. 10]. The Split Linear Fit smoother produces piece-wise smooth curves and thus will depict discontinuities present in the input data [Ref. 10: p. 1]. Most smoothers tend to distort discontinuities because the weighted averaging technique used to compute a smoothed point requires a continuous underlying function. The Split Linear Fit smoother will not distort the smooth curve at discontinuous points and does a very good job of detecting sharp slopes in the input data. This is the reason the Split Linear Fit algorithm is sometimes classified as an edge-detecting smoother [Ref. 10: p. 2].

The Split Linear Fit smoother is similar to Friedman and Stuetzle's Supersmoother in several ways:

1. every input point receives a respective smooth point;
2. the user can enter more than one neighborhood size; in this algorithm the neighborhood sizes are called window sizes, where window of size K is defined as "a set of K successive point" [Ref. 10: p. 2], (window size and span are equivalent terms);
3. the window is shifted to the right by dropping the left endpoint and then adding the point adjacent to the right endpoint;
4. the method of least squares is used to estimate a straight line through the points within the shifting window;
5. the Split Linear Fit smoother is scale independent.

A major difference between the Split Linear Fit smoother and Supersmoother is the method used to combine the linear fitted values. Another difference is that the Split linear Fit smoother does only robust smoothing.

The objective of the Split Linear Fit smoother is to produce a piecewise smoothed curve with minimal discontinuous features [Ref. 10: p. 2]. Figure 3.1 shows the Split Linear Fit smoothing algorithm as composed of three subroutines:

1. the Regression Subroutine;
2. the Weighting Subroutine;
3. the Combining Subroutine.

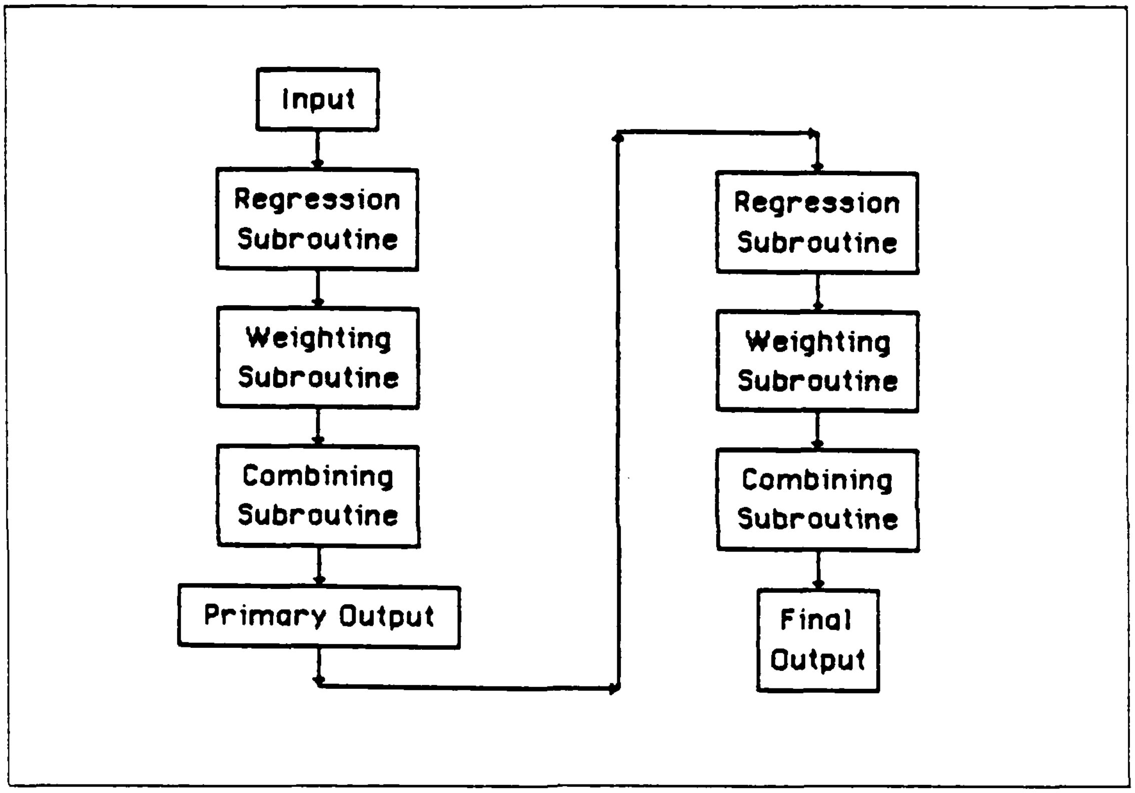


Figure 3.1 Data Flow in Split Linear Fit Smoother.

Figure 3.1 also shows that the Split Linear Fit smoother uses the iterative process once on its output. This is done in order to decrease the variance in the first set of output, since, as mentioned before, the Split Linear Fit smoother only does robust smoothing. If the first set of output were to be plotted, the curve would appear very jagged. Passing the first set of output through the Split Linear Fit algorithm, decreases the robustness and variability of the final output.

The number of window sizes entered by the user dictates the number of times that the input data is passed through the Regression Subroutine to produce a family of linear fitted values and residual values associated with each I, I=1, . . ., N, where N is the number of points to be smoothed, see Figure 3.2. Each family of linear fits may be viewed as a pseudo-distribution of linearly fitted estimated values of an input point value.

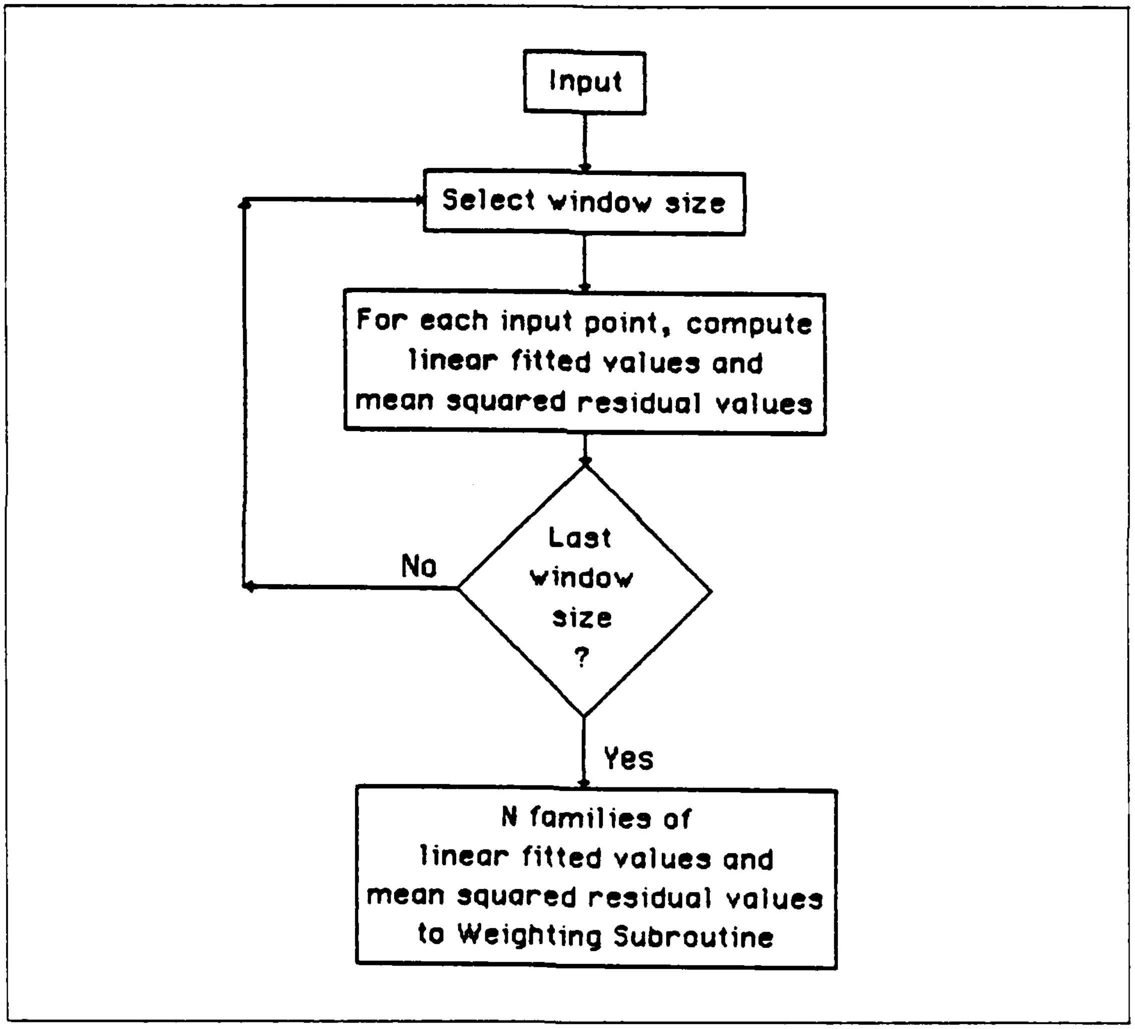

Input
Select window size
For each input point, compute linear fitted values and mean squared residual values
Last window size ?
No
Yes
N families of linear fitted values and mean squared residual values to Weighting Subroutine

Figure 3.2 Regression Subroutine in Generalized Form.

The Weighting Subroutine receives the N families of linear fits and mean squared residual values from the Regression Subroutine and finds the minimum mean squared residual value within each family of mean squared residual values. Not all the mean squared residual values qualify as candidates for the minimum mean squared residual values. An acceptable mean squared residual value is one that does not exceed an established cutoff value. The cutoff value used in the Split Linear Fit algorithm is the value -1.0×10^{30}. This value was selected because it provided a better smooth curve at discontinuities inherent in the raw, input data than other cutoff values [Ref. 10: p. 3]. The minimum mean squared residual value is used as a base to compute a weight corresponding to each of the acceptable mean squared residual values within the family, see Figure 3.3. These weights are used by the Combining Subroutine in computing the smooth point values. The weights "depend on a measure of the quality of the corresponding linear fits" [Ref. 10: p. 2]. Quality meaning that the smaller the mean squared residual value the higher the weight assigned to the corresponding fitted value. The weight assigned to a fitted value is a function of the following:

1. the corresponding mean squared residual;
2. the minimum mean squared residual, and;
3. the average of the acceptable mean squared residuals within the associated window.

This weighting procedure is used in order to smoothly integrate discontinuities in the input data with the other smooth points. This procedure is the edge-detector and is the cause of the robust smoothing.

The smooth point value at I is a weighted average of the linear fits in the family of linear fits corresponding to I. The Combining Subroutine combines the weights produced by the Weighting Subroutine and the fitted values produced by the Regression Subroutine associated with I and computes the respective smooth point value, see Figure 3.4.

As mentioned before the first set of smooth point values produced by the Combining Subroutine is itself passed through the Split Linear

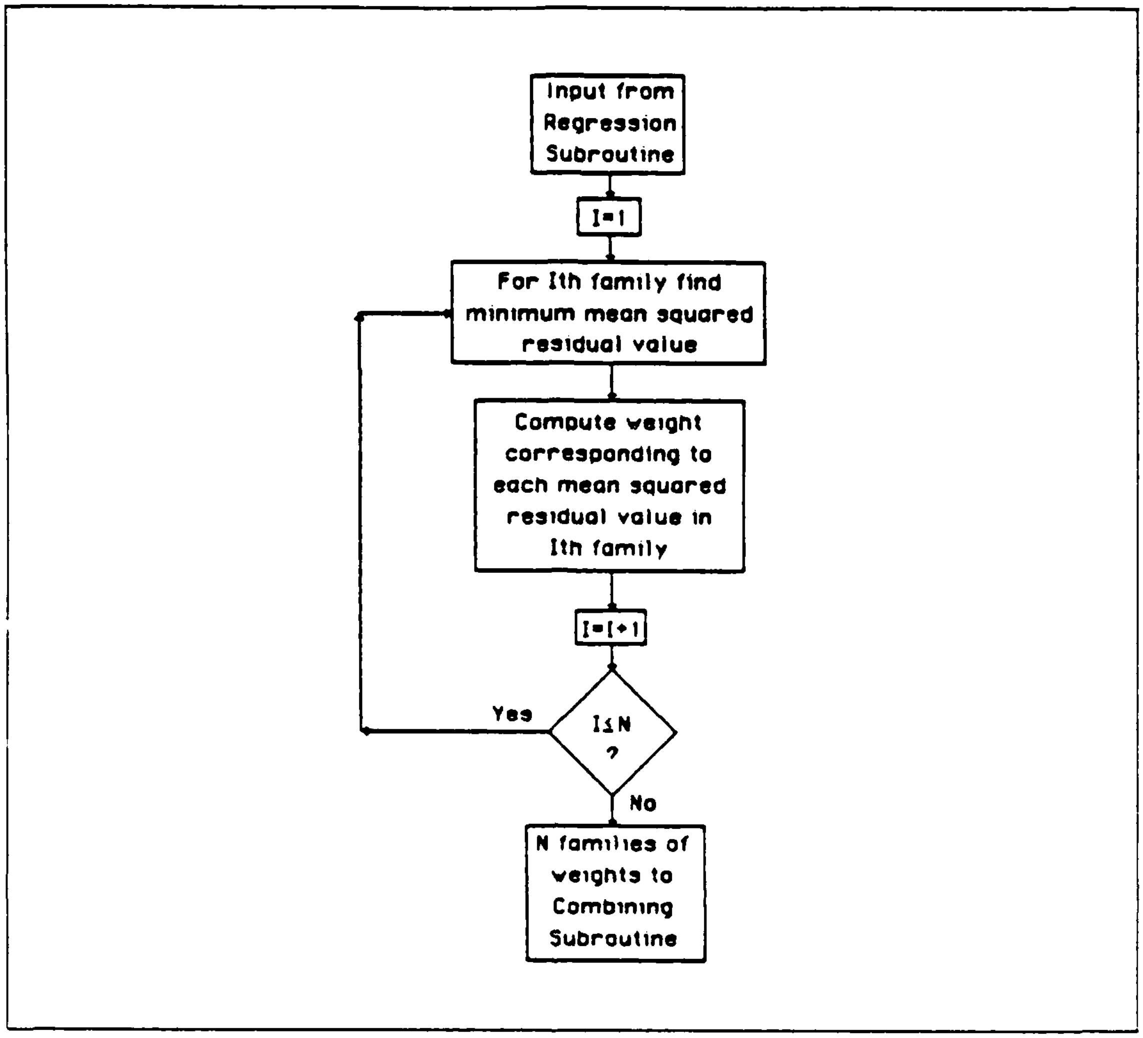

Input from Regression Subroutine
I=1
For Ith family find minimum mean squared residual value
Compute weight corresponding to each mean squared residual value in Ith family
I=I+1
I≤N ?
Yes
No
N families of weights to Combining Subroutine

Figure 3.3 Weighting Subroutine in Generalized Form.

Fit algorithm again in order to reduce the variability. The second set of smooth point values is the output generated to the user, see Figure 3.1.

B. MATHEMATICAL DETAILS---REGRESSION SUBROUTINE

The Regression Subroutine requires the following user input:

1. N, the number of points to be smoothed;
2. Y_1, . . ., Y_N, the point values to be smoothed (in chronological order);

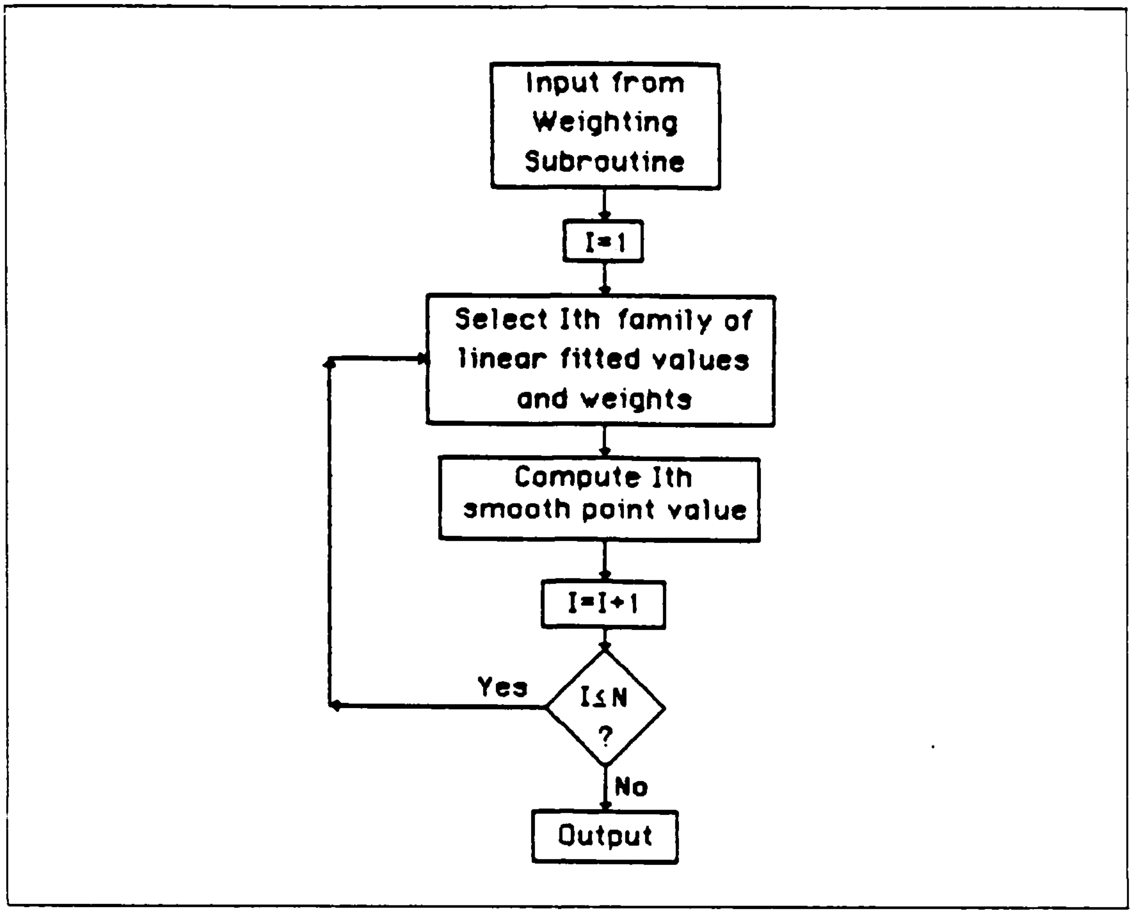

Input from Weighting Subroutine
I=1
Select Ith family of linear fitted values and weights
Compute Ith smooth point value
I=I+1
Yes
I≤N ?
No
Output

Figure 3.4 Combining Subroutine in Generalized Form.

3. X_1, . . ., X_N, the abscissa corresponding to the Y values (in ascending order since the Y_I are in chronological order);
4. NTRYS, the number of window sizes to be used;
5. $WNSZ_1$, . . ., $WNSZ_{NTRYS}$, the values of the window sizes;
6. MNWNSZ, the minimum window size permitted by the user.

The minimum window size, MNWNSZ, is the lower bound set on the window size. The value of the lower bound should be at most one-half the value of the smallest window that will be used in the smoothing. If MNWNSZ is any larger then some smooth points will be dropped from the ends of the output array, or a plot of the smooth point values will show distortion at the ends.

Figure 3.2 shows the Regression Subroutine using a procedure resembling the iterative process, but each pass of the Y_I values through the Regression Subroutine uses a different window size, and some variables are reset to zero. The purpose of the Regression Subroutine is to compute a family of fitted values and a family of mean squared residual values corresponding to each I, I= 1, . . ., N. The Regression Subroutine is shown in more detail in Figure 3.5. The Regression Subroutine can be divided into three parts:

1. definition of zero and computation of sum of first (MNWNSZ-1) values and computation of fitted values for I=1, . . ., (MNWNSZ-1);
2. shifting of window and computation of fitted values and mean squared residual values for I=MNWNSZ, . . ., (N-MNWNSZ+1);
3. computation of fitted values for I=(N-MNWNSZ+2), . . ., N.

The variable EPS is used to define zero for computational purposes. The interquartile range of the abscissa array is used in the computation of EPS, as shown by equations 3.1 through 3.3:

$$JL = \frac{N}{4} \; ; \tag{3.1}$$

$$JR = 3 \times JL \; ; \tag{3.2}$$

$$EPS = X_{JR} - X_{JL} \; . \tag{3.3}$$

If EPS≤0.0 and JR<N, then EPS is recomputed using the following three rules:

1. if JR<N, then JR is increased by a value of one;
2. if JL>1, then JL is decreased by a value of one;
3. EPS is recomputed using the new values of JR and JL and equation 3.3.

EPS will be equal to zero only if N≤3 and if the computer allows index values equal to zero, otherwise a computer error will result. If this situation occurs then items 1 and 3 from above will apply. Since the X_I values are in ascending order, EPS will have a value greater than zero after the first recomputation of JR. After EPS has been defined as being greater than zero, it is adjusted using equation 3.4 in order to define zero for computational purposes:

$$\mathrm{EPS} = (EPS \times 1.0\times10^{-10})^2 \; . \tag{3.4}$$

In order to fit a linear model on the window endpoints and central point, slope, and y-intercept values of the model must first be computed. The parameter MNWNSZ dictates which input point values will be used to compute the initial slope and the corresponding y-intercept value. The first MNWNSZ values of the input data are used to compute these necessary values. The first (MNWNSZ-1) values of the input data are used in equations 3.5 through 3.10 to compute the basic sum values to be used later and increment a counter which keeps track of the input points:

$$\mathrm{SUM}_X = \sum_{I=1}^{\mathrm{MNWNSZ}-1} X_I \; ; \tag{3.5}$$

$$\mathrm{SUM}_Y = \sum_{I=1}^{\mathrm{MNWNSZ}-1} Y_I \; ; \tag{3.6}$$

$$\mathrm{KOUNTER} = \mathrm{MNWNSZ} - 1 \; ; \tag{3.7}$$

$$\mathrm{SUM}_{XSQ} = \sum_{I=1}^{\mathrm{MNWNSZ}-1} X_I^2 \; ; \tag{3.8}$$

$$\mathrm{SUM}_{YSQ} = \sum_{I=1}^{\mathrm{MNWNSZ}-1} Y_I^2 \; ; \tag{3.9}$$

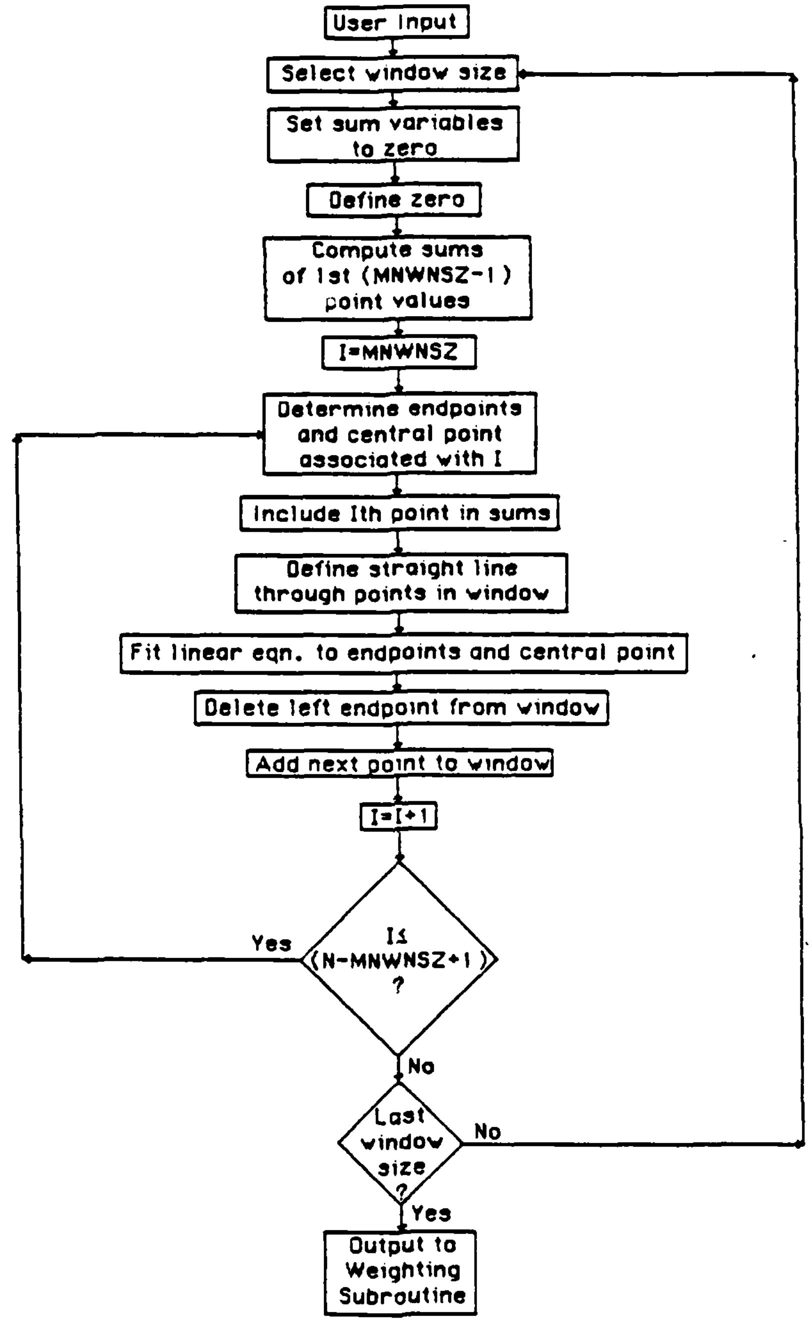

User Input
Select window size
Set sum variables to zero
Define zero
Compute sums of 1st (MNWNSZ-1) point values
I=MNWNSZ
Determine endpoints and central point associated with I
Include Ith point in sums
Define straight line through points in window
Fit linear eqn. to endpoints and central point
Delete left endpoint from window
Add next point to window
I=I+1
Is I≤ (N-MNWNSZ+1) ?
Yes
No
Last window size ?
No
Yes
Output to Weighting Subroutine

Figure 3.5 Data Flow in the Regression Subroutine.

$$SUM_{XY} = \sum_{I=1}^{MNWNSZ-1} (X_I \times Y_I) \ . \qquad (3.10)$$

For each I=1, . . ., (MNWNSZ-1), the right endpoint fitted value and the mean squared residual value are set to the value of -1.0×10^{30}. This is done so that the Weighting Subroutine will assign a zero weight to the left endpoint fitted values and the mean squared residual values corresponding to these I. Therefore, the smooth values corresponding to these I values are computed using a smaller window [Ref. 10: p. 3]. This procedure does not use the window concept that is mentioned below. This is done to avoid computer errors since most of the points in the window corresponding to these I will have negative index values.

Figure 3.5 shows the iterative process that is used to compute the family of fitted values and the family of mean squared residual values corresponding to I, I=MNWNSZ, . . ., (N-MNWNSZ+1). The first step determines the window central point and the endpoints that correspond to I. For ease of understanding, let K be the number of successive points in the window. If K is odd then the central point is equivalent to the median of the window and has $\lfloor(K/2)$ neighboring points to the left and right of it. If K is an even number, then the central point is will have [(K/2)-1] neighboring points to the left and (K/2) neighboring points to the right, thus the central point is the point to left of the window median. The index of the right endpoint will always be equal to I. If K is odd, the index of the central point will be equal to [I- $\lfloor(K/2)$], and the index of the left endpoint will be equal to (I-K+1). If the value of K is even, the index of the central point will be equal to [I-(K/2)], and the index of the left endpoint will be equal to (I-K+1). Point values that have corresponding index values that are negative or zero are not included in the linear fit.

The next step adds the I^{th} point to the window and uses the method of least squares to estimate the straight line through the points in the window. The procedure for adding the I^{th} point to the neighborhood adds the values produced by equations 3.5 through 3.10 to the X_I and the Y_I values using equations 3.11 through 3.16:

$$SUM_X = SUM_X - X_I \; ; \qquad (3.11)$$

$$SUM_Y = SUM_Y - Y_I \; ; \qquad (3.12)$$

$$KOUNTER = KOUNTER + 1 \; ; \qquad (3.13)$$

$$SUM_{XSQ} = SUM_{XSQ} - X_I^2 \; ; \qquad (3.14)$$

$$SUM_{YSQ} = SUM_{YSQ} + Y_I^2 \; ; \qquad (3.15)$$

$$SUM_{XY} = SUM_{XY} + (X_I \times Y_I) \; . \qquad (3.16)$$

Next the mean of the sum values computed by equations 3.11 through 3.16 is computed using the value of KOUNTER as the denominator in equations 3.17 through 3.21:

$$MEAN_X = \frac{SUM_X}{KOUNTER} \; ; \qquad (3.17)$$

$$MEAN_Y = \frac{SUM_Y}{KOUNTER} \; ; \qquad (3.18)$$

$$MEAN_{XSQ} = \frac{SUM_{XSQ}}{KOUNTER} \; ; \qquad (3.19)$$

$$MEAN_{YSQ} = \frac{SUM_{YSQ}}{KOUNTER} \; ; \qquad (3.20)$$

$$MEAN_{XY} = \frac{SUM_{XY}}{KOUNTER} \; . \qquad (3.21)$$

The variance of the abscissa is derived by equation 3.22:

$$XVAR = MEAN_{XSQ} - MEAN_X^2 \; . \qquad (3.22)$$

The method of least squares is used to compute the slope and the y-intercept of the straight line fitted to the points in the window. The results produced by equations 3.17 through 3.18 are used to compute the coefficients of the straight line that is fitted to the points within the window. If XVAR≤0.0, then the slope of the straight line, SLOPE, is zero, i.e. SLOPE = 0.0, otherwise the value of SLOPE is computed with equation 3.23:

$$SLOPE = \frac{MEAN_{XY} - (MEAN_X \times MEAN_Y)}{XVAR} . \qquad (3.23)$$

The y-intercept of the straight line is computed using equation 3.24:

$$INTER = MEAN_Y - (SLOPE \times MEAN_X) . \qquad (3.24)$$

The mean squared residual value about the fitted line is computed with equations 3.25 through 3.28:

$$MEAN_{RSQ} = A + B + C ; \qquad (3.25)$$

where

$$A = MEAN_{YSQ} - (2 \times INTER \times MEAN_Y) - (2 \times SLOPE \times MEAN_{XY}) ; \qquad (3.26)$$

$$B = INTER^2 + (2 \times INTER \times SLOPE \times MEAN_X) ; \qquad (3.27)$$

$$C = MEAN_{XSQ} \times SLOPE^2 . \qquad (3.28)$$

The window central point and endpoints are fitted to a linear model using the slope and the y-intercept value computed above to produce the fitted value FIT_{IWP}, where I= current I^{th} value, W= current window size, and P= left endpoint, central point, or right endpoint. The mean squared residual value, $MSQR_{IWP}$, is computed using the computed local linear fit coefficients, the X_I and Y_I values, and the counter value in equations 3.29 through 3.31:

$$FIT_{IWP} = INTER + (SLOPE \times X_I) ; \qquad (3.29)$$

$$RES = Y_I - FIT_{IWP} ; \qquad (3.30)$$

$$MSQR_{IWP} = \frac{(KOUNTER \times MEAN_{RSQ}) - RES^2}{KOUNTER - 1} \quad . \tag{3.31}$$

After FIT_{IWP} and $MSQR_{IWP}$ have been computed for the I^{th} window's central point and endpoints, the window must be shifted to the right. The Supersmoother algorithm uses the same procedure as the Split Linear Fit, i.e. dropping the left endpoint from the sum values of equations 3.11 through 3.16 and then adding the entering point to these same equations. Let IL be the index of the left endpoint, then this point is dropped using equations 3.32 through 3.37:

$$SUM_X = SUM_X - X_{IL} \quad ; \tag{3.32}$$

$$SUM_Y = SUM_Y - Y_{IL} \quad ; \tag{3.33}$$

$$KOUNTER = KOUNTER - 1 \quad ; \tag{3.34}$$

$$SUM_{XSQ} = SUM_{XSQ} - X_{IL}^2 \quad ; \tag{3.35}$$

$$SUM_{YSQ} = SUM_{YSQ} - Y_{IL}^2 \quad ; \tag{3.36}$$

$$SUM_{XY} = SUM_{XY} - (X_{IL} \times Y_{IL}) \quad . \tag{3.37}$$

Next, I is incremented by one and, if $I \leq (N\text{-}MNWNSZ+1)$, equations 3.11 through 3.16 are used to enter the new I^{th} point into window. Equations 3.17 through 3.37 are then repeated using the new values. This procedure is continued until $I > (N\text{-}MNWNSZ+1)$.

When $I > (N\text{-}MNWNSZ+1)$, the left endpoint FIT_{IWP} and $MSQR_{IWP}$ values corresponding to the values of I are set equal to -1.0×10^{30}, so that these fitted values are assigned no weight in the Weighting Subroutine. This procedure was used for I= 1, ..., (MNWNSZ-1) at the beginning of the Regression Subroutine.

If the user entered more than one window size, then the input data is passed through the Regression Subroutine with the next window size.

The sums and counter of equations 3.5 through 3.10 are initialized to zero before repeating the Regression Subroutine. After the last window size has been used, each window will have contributed a total of six values to each I, i.e. three linear fitted values and three mean squared residual values.

C. MATHEMATICAL DETAILS---WEIGHTING SUBROUTINE

The objective of the Weighting Subroutine is to compute a weight which is indicative of the degree of goodness of fit of each linear fitted value, FIT_{IWP}, with respect to a line with slope equal to one. Figure 3.3 shows the procedure followed by the Weighting Subroutine. As noted in the figure, each family of mean squared residual values is used one set at a time. The following data is transferred from the linear Regression Subroutine:

1. N,the number of points to be smoothed;
2. NTRYS,the number of windows used in the smoothing;
3. N families of mean squared residual values, $MSQR_{IWP}$;
4. N families of fitted values, FIT_{IWP}.

The Weighting Subroutine is executed once for each family of mean squared residual values. The Regression Subroutine produced a family of (3×NTRYS) mean squared residual values corresponding to each I. For computational feasibility a lower bound of -1.0×10^{30} is set on the values of mean squared residual, i.e. the $MSQR_{IWP}$. Each $MSQR_{IWP}$ value is compared against -1.0×10^{30}, and the $MSQR_{IWP}$ values less than or equal to -1.0×10^{30} are marked as unacceptable. These are not considered in the search for the minimum $MSQR_{IWP}$ within the I^{th} family $MSQR_{IWP}$. The minimum $MSQR_{IWP}$ corresponding to I is found by doing a comparison between the acceptable $MSQR_{IWP}$ values in the I^{th} family of $MSQR_{IWP}$. The expressions listed below are used by the Weighting Subroutine on each family of $MSQR_{IWP}$:

1. MIN is the minimum $MSQR_{IWP}$ in I^{th} family;
2. LAMBDA is the sum of $MSQR_{IWP}$ greater than -1.0×10^{30};
3. LAMBDA is divided by the number of $MSQR_{IWP}$ greater than -1.0×10^{30} ;

4. LAMBDA is then reduced by the value of MIN.

If the value of LAMBDA is less than or equal to zero, then LAMBDA is not modified, otherwise LAMBDA is recomputed using equation 3.38:

$$\mathrm{LAMBA} = \frac{1.0}{\mathrm{LAMBA}} \quad (3.38)$$

If the value of MIN is positive, MIN is not changed, otherwise it is made a positive value using equation 3.39:

$$\mathrm{MIN} = 1\times 10^{-10} \quad (3.39)$$

The last step in the Weighting Subroutine is to compute a family of weights which indicate the goodness of fit of the linear fitted values, $\mathrm{FIT}_{\mathrm{IWP}}$, using the corresponding family of $\mathrm{MSQR}_{\mathrm{IWP}}$ values. The $\mathrm{MSQR}_{\mathrm{IWP}}$ values which are less than -1.0×10^{30} cause a weight of zero to be attached to the corresponding fitted value, $\mathrm{FIT}_{\mathrm{IWP}}$.. The reason for this occurring is that these values are considered unacceptable based on the established cutoff value discussed in Section A of this chapter. If the $\mathrm{MSQR}_{\mathrm{IWP}}$ value satisfies the cutoff rule, then a weight will be computed indicating the goodness of fit of the corresponding $\mathrm{FIT}_{\mathrm{IWP}}$. The weight is a function of the quality of the corresponding $\mathrm{MSQR}_{\mathrm{IWP}}$ value. TEMP indicates the degree of quality and is computed using equation 3.40:

$$\mathrm{TEMP} = \mathrm{LAMBA}\cdot(\mathrm{MSQR}_{\mathrm{IWP}} - \mathrm{MIN}) \quad (3.40)$$

Recall that the smaller the value of $\mathrm{MSQR}_{\mathrm{IWP}}$, the better the value of $\mathrm{FIT}_{\mathrm{IWP}}$. In other words, small values of TEMP indicate a good $\mathrm{FIT}_{\mathrm{IWP}}$ value, therefore, these fit values receive high weights. Three conditions are used in assigning a weight that to each acceptable $\mathrm{MSQR}_{\mathrm{IWP}}$:

1. if $\mathrm{TEMP}\le 0.0$, then $\mathrm{WT}_{\mathrm{IWP}} = 1.0$;
2. if $0.0<\mathrm{TEMP}<1.0$, then the weight is computed using equation 3.41:

$$WT_{IWP} = (1.0 - TEMP)^2 \ ; \qquad (3.41)$$

3. if TEMP≥1.0, then $WT_{IWP} = 0.0$.

According to the Split Linear Fit developer, equation 3.41 results in "smooth transitions between zero weights and small non-zero weights" while other functions tend not to have the desired effect [Ref. 10: p. 3]. Recall that the Weighting Subroutine is repeated for each family of mean squared residual values and that the output is a family of weights corresponding to each I.

D. MATHEMATICAL DETAILS---COMBINING SUBROUTINE

The objective of the Combining Subroutine is to compute the smooth point values. The following data is transferred from the Weighting Subroutine:

1. N, the number of points to be smoothed;
2. NTRYS, the number of bandwidths used in the smoothing;
3. N families of fitted values, FIT_{IWP};
4. N families of weights, WT_{IWP}.

Before using a FIT_{IWP} value in the following computations, it is compared to the value -1.0×10^{30}. FIT_{IWP} values less than or equal to -1.0×10^{30} are marked as unacceptable and not used in the computations of the corresponding smooth value. Using each family of WT_{IWP} and FIT_{IWP}, the Combining Subroutine computes a weighted average of the linear fitted values, FIT_{IWP}. The first step in computing the I^{th} smooth point is to use the corresponding family of FIT_{IWP} and WT_{IWP} values in equations 3.42 and 3.43:

$$RSUM = \sum_{W}\sum_{P} FIT_{IWP}\,, \quad \text{for I}= 1,2,\ldots N \ ; \qquad (3.42)$$

$$WSUM = \sum_{W}\sum_{P} WT_{IWP}\,, \quad \text{for I}= 1,2,\ldots N \ ; \qquad (3.43)$$

If $WSUM\geq1.0\times10^{-10}$, then the I^{th} smooth point is produced by equation 3.44, otherwise $SMOOTH_I$ equals -1.0×10^{30}.

$$\text{SMOOTH}_I = \frac{\text{RSUM}}{\text{SUM}} \ . \tag{3.44}$$

Next, I is incremented by one, and the Combining Subroutine is repeated using the new value of I and the corresponding families of FIT_{IWP} and WT_{IWP} values. This procedure continues until I>N. If these smoothed point values were plotted, the plot would show too many peaks and would appear very jagged. In order to alleviate this problem, these smoothed point values are used as input data to the Regression Subroutine, and the Split Linear Fit algorithm is executed one more time. The output of the second pass does not have as much variability as the output from the first pass and is thus more useable for data reduction.

E. SELECTION OF WINDOW SIZE

The section on selection of span value in the previous chapter applies in this chapter to a great degree. Window size and span value are equivalent terms, and both affect the smoothing to a great degree. Since both the Supersmoother and the Split Linear Fit use local linear regression, the relationship between a window size and the degree of smoothing can be explained by equation 1.4. The effect on the residual values caused by varying the window size in equation 1.4 must be kept in mind when selecting a window size, i.e. remember the following:

1. large window sizes produce a smooth plot;
2. small window sizes produce a not so smooth plot.

What equation 1.4 is illustrating is that the degree of smoothness is the result of a tradeoff between bias and variance in the resulting smooth plot since it is an estimation of a function in the presence of additive errors [Ref. 10: p. 1]. A satisfactory trade-off between bias and variance is difficult to obtain. Better decisions can be made by looking at a plot of the smooth output. Therefore, the user of a smoothing program should plot the smooth output and then decide if the results satisfy his/her needs and desires, otherwise the smoothing program is run again with a different window size. Some people want and need very 'smooth' and highly biased results while others want results on the other extreme, i.e low bias and high variance.

The above paragraph explains in general the effects produced by using one window size, but the Split Linear Fit smoother can accept more than one window size. By using more than one window size the desired smooth output may be found in less time than if a single window size had been used. The reason for this speed is that the Split Linear Fit algorithm has more information about the shape of the raw data, than when only one window size is used, i.e. a pseudo-distribution of fitted values about each raw point is produced. Then the smoothed point is computed using this additional information, i.e. a weighted-average of the fitted values. But the user needs to apply the basic relationship between window size and degree of smoothness stated in the above paragraph in selecting a set of window sizes, i.e. a set of 'large' windows will produce a smoother effect than a set of 'small' windows.

According to McDonald and Owen it is best to use a set of three to five consecutive odd window sizes [Ref. 10: pp. 2-4]. A mixture of small and large window sizes will result in centrally smooth point values with a slight degree of variability. In order to be able to accurately trace the curvature of the input data, it is best to do the following:

1. roughly measure the periodicity of the input data;
2. use this value as one of the window sizes to be used in the smoothing;
3. select the other window sizes with respect to the value of the periodicity.

For example, if the periodicity of the input data is estimated to be 27, then 27 is used as an input window size and the other window sizes may be 23, 25, 29, and 31. Or the periodicity value may be either the smallest window size or the largest window size while the other window sizes are selected with respect to the periodicity.

The other factor that has great influence on the smooth output produced by the Split Linear Fit is the minimum window size, MNWNSZ. If this value is too large then the smooth output will not be what is expected. It is best to keep the value of MNWNSZ at no more than one-half the value of the smallest window. This subject will be discussed in more detail in the evaluation of the Split Linear Fit smoother.

IV. EVALUATION OF THE ADVANCED SMOOTHING ALGORITHMS

A. GENERAL

As stated in Chapter I, a smoothing algorithm may be compared to a low-pass filter which is designed to do the following:

1. filter out "noise" from a data set and;
2. estimate the underlying functional relationship present in the given data set.

Before proposing a 'good and efficient' smoothing algorithm to an individual, the user must be shown that the 'good and efficient' smoothing algorithm is robust. In other words, it is necessary to illustrate that the smoothing algorithm performs well with most data sets, whether the underlying function is either of the following:

1. simple like a linear function or a simple trigonometric function, or;
2. complex and is very difficult to define mathematically.

In this chapter, the input data sets used with the Supersmoother and the Split Linear Fit smoothing algorithms are generated from simple known functions with Normal (0,1) "noise" added. The GRAFSTAT [Ref. 12] functions used to generate the pseudo-random Normal (0,1) deviates are given in Section 1 of Appendix C. The output produced by these algorithms is evaluated in order to do the following:

1. observe how well the Normal (0,1) "noise" is filtered by the smoother, and;
2. determine how well the true function is extracted and depicted.

In this chapter the input data sets have a constant variance of 1. In the next chapter the input data set dose not necessarily have a constant variance, because it is real, and the unknown underlying function is probably too complex to define succinctly.

B. METHODOLOGY

There is no established procedure to follow in testing the efficiency and effectiveness of a smoothing technique/algorithm. Since the adequacy or usefulness of he smooth output is largely based on the user's subjective judgement of the shape of the smooth curve, i.e. a plot of the smooth point values, the most effective method is to compare the output produced by the new algorithm to the output produced by previously validated, widely used, and well liked smoothing algorithms, e.g. LOWESS [Ref. 3: pp. 94-104]. The following procedure is used to evaluate the Supersmoother and the Split Linear Fit smoothing algorithms:

1. explain and display the data set to be smoothed;
2. display and explain the smooth results produced by the baseline smoothing techniques/algorithms, i.e. Least Squares Regression, Equal-Weight Moving Average, Cosine-Weighted Moving Average, and LOWESS;
3. display and examine the smooth results produced by the advanced smoothers;
4. compare these results to the previously discussed results from 2 above.

The GRAFSTAT graphics package [Ref. 12] was used to generate all of the graphs in this thesis. GRAFSTAT was also used to do the Least Squares Regression and the Equal-Weight Moving Average smoothing.

The Method of Least Squares tries to standardize the curves that can be fitted to a data set. The measure of performance that is used with this global fitting technique is the sum of squared residuals. The Method of Least Squares produces a smooth curve which closely approximates the given set of data points and which minimizes the sum of squared residuals attainable with the chosen global curve. For more explicit details see Spiegel [Ref. 13: pp. 258-305]. GRAFSTAT lets the user select the type of curve that should be fitted to the given data set. The following listed curve fits which use the Method of Least Squares were used in this thesis and are available in the GRAFSTAT graphics package:

1. linear curve fit;
2. quadratic curve fit;
3. third degree polynomial curve fit, and;
4. Spline fit.

Least Squares Regression with linear fit is a technique of finding the linear equation which fits a data set and minimizes the sum of squared residuals. Least Squares Regression with quadratic fit does the same, but the data is fitted to an second degree polynomial equation, i.e. $Y = AX^2 + BX + C$, where A, B, and C are the estimated coefficients, X is the abscissa corresponding to the data being smoothed, and Y is an estimate of the data being smoothed. Least Squares Regression with third degree polynomial curve fit is also basically the same as previous two techniques, but the equation being fitted to the given data has the form of $Y = AX^3 + BX^2 + CX + D$, where A, B, C, and D are the estimated coefficients and X and Y are the same as for the quadratic fit. For more details about Least Squares Regression with either linear fit, quadratic fit, or third degree polynomial fit see Spiegel [Ref. 13: pp. 258-305]. All of these techniques use global curve fitting, i.e. the curve is fitted to the given data as an entity. The Spline fit on the other hand uses local curve fitting in order to produce the smoothest possible curve with the sum of squared residuals value less than or equal to a parameter entered by the user [Ref. 12]. The Spline curve fitting technique uses the Least Squares Method with third degree polynomial within a predetermined neighborhood of the given data. The neighborhood size is predetermined by the developers of this GRAFSTAT function. The second derivative of the defined cubic equation is computed and evaluated using the median of the neighborhood of point values. The neighborhood is then shifted to the next point and the procedure is repeated. The sum of squared residuals is computed and compared to the maximum sum of squared residuals value that the user requires. If this value exceeds the users constraint then the entire procedure is repeated. In other words, the Spline curve fitting technique is a constrained linear programming problem, where the constraint is the user's maximum sum of squared residuals value desired, and the objective function is a cubic equation [Ref. 16: pp. 77-87]. The user of these curve fitting technique should first look at a scatterplot of the raw data and then decide which one to use. Therefore, if a scatterplot of a data set looks linear, the user should attempt to fit a linear curve to the data, otherwise, one of the other curve fitting techniques should be used.

The Equal-Weight Moving Average smoother was briefly discussed in Chapter I during the discussion of equations 1.3 and 1.4. The Equal-Weight Moving Average GRAFSTAT functions that were used to generate the smooth point values are shown in Appendix C. The following equation defines the smooth points produced by the Equal-Weight Moving Average smoother:

$$s(X_{I+\frac{K-1}{2}}) = \sum_{J=I}^{I+K} \left(\frac{1}{K} \times Y_J \right) \text{ . } I = 1,2,\ldots,(N-K), \qquad (4.1)$$

where N is the number of points to be smoothed and K is the neighborhood size, i.e. the number of points involved in the averaging. Both N and K must be positive, non-zero integers, with K being odd. For an expansion of the Equal-Weight Moving Average smoothing theory, see Anscombe [Ref. 14: pp. 153-159].

The Cosine Weighted Moving Average smoother is an extension of the Equal-Weight Moving Average smoother. Instead of using the inverse of K as the weight for each Y value within the neighborhood, a cosine related weight is computed for each of the Y values within the neighborhood of size K. (The APL functions used to generate these values appear in Appendix C.) These cosine weights are a function of the Y values' location within the shifting window/neighborhood of size K. The expression defining the smoothed output of the Cosine-Weighted Moving Average smoother is:

$$s(X_{I+\frac{K-1}{2}}) = \sum_{J=1}^{K} \left(WT_J \times Y_{I+J-1} \right) \text{ , } I = 1,2,\ldots,(N-K), \qquad (4.2)$$

where .

Anscombe [Ref. 14: p. 450] characterizes this smoother as a "better-quality approximation" than other Moving Average smoothers, because "not only does the integrand, but also its first derivative vanish at both ends of the range of integration" [Ref. 14: pp. 156-157].

The last smoother used as a base against which the advanced smoothers were tested is the Locally Weighted Regression Scatter Plot Smoother, commonly called LOWESS [Ref. 3: pp. 94-104]. LOWESS uses the smoothing technique referred to as local regression, i.e. the Method of Least Squares is used on the input points within a user given neighborhood. Only one neighborhood size is used by LOWESS per run of the program. The program requires that the user not enter the neighborhood size to be used, but a parameter called F, which is a ratio of the neighborhood size to the number of points to be smoothed. The user has the option of fitting either a linear or a quadratic function to the point values within the neighborhood. In addition, the user has the option of using robust or non-robust smoothing. Robust smoothing has more variability than non-robust smoothing, because outliers are emphasized. Each input point within the neighborhood receives a weight which is a function of the point's location with respect to the median of the neighborhood. These weighted point values are then used to define a fitted curve within the neighborhood of input point values. The coefficients of the defined curve and the median of the neighborhood are used to compute the smoothed point value corresponding to the median of the neighborhood. The neighborhood size is shifted from one point to the next until each input point has a corresponding smoothed point value.

Each smoother being used in this thesis requires that a neighborhood be indicated by the user, but each of these smoothers calls the neighborhood size by a different name as discussed in this section. The term 'neighborhood size', i.e. the number of point values involved in the averaging, can be used by any of the smoothers being discussed in this thesis. The Moving Average smoothers use the variable M to indicate the neighborhood size. The LOWESS smoother uses the variable F, as discussed the above paragraph. Supersmoother uses a value called SPAN which is equivalent to the F value of LOWESS. The Split Linear Fit smoother uses the term 'window size' which is equivalent to the M value of the Moving Average smoothers.

C. TESTING AND RESULTS----LINEAR UNDERLYING FUNCTION

The first test posed on the Supersmoother and the Split Linear Fit algorithms was to detect linear trends in a data set which does have a linear trend. Figure 4.1 shows Test Set One which consists of 200 data points produced from the following equation:

$$Y = X + \text{Normal}\,(0,1)\ \text{noise},\ 0<X\leqslant 200\ . \qquad (4.3)$$

The values produced by this function are in tabular form in Appendix D. Figure 4.2 shows the results from doing a linear regression on Test Set One. It is obvious that the linear regression curve and the true linear curve do not coincide. A Confidence Interval Test on the coefficients produced by the linear regression reveals that the Y-intercept coefficient, 0.0023573 is not significantly different from zero with a Confidence Level greater than 0.8. The slope coefficient has a Confidence Level less than 0.001 that it is not significantly different from zero. Therefore, the linear regression curve can be reduced to $Y = 1.0104X$ which has a standard deviation of 0.031 which includes the true linear relationship, $Y = X$.

The LOWESS smoothing results are shown in Figure 4.3. Since Test Set One appears to be linear, the linear option of LOWESS is used. There is little visible difference between the results produced by using the robust option and the results produced by using the non-robust option, i.e. compare the left-hand smooth plots with the right-hand smooth plots of Figure 4.3. This is not surprising since there are no outliers in this artificial data. The graphs in Figure 4.3 show that as the F value is increased, the curve produced gets smoother, i.e. as the neighborhood size increases the curve gets smoother because the bias increases and the variance decreases. All F values greater than or

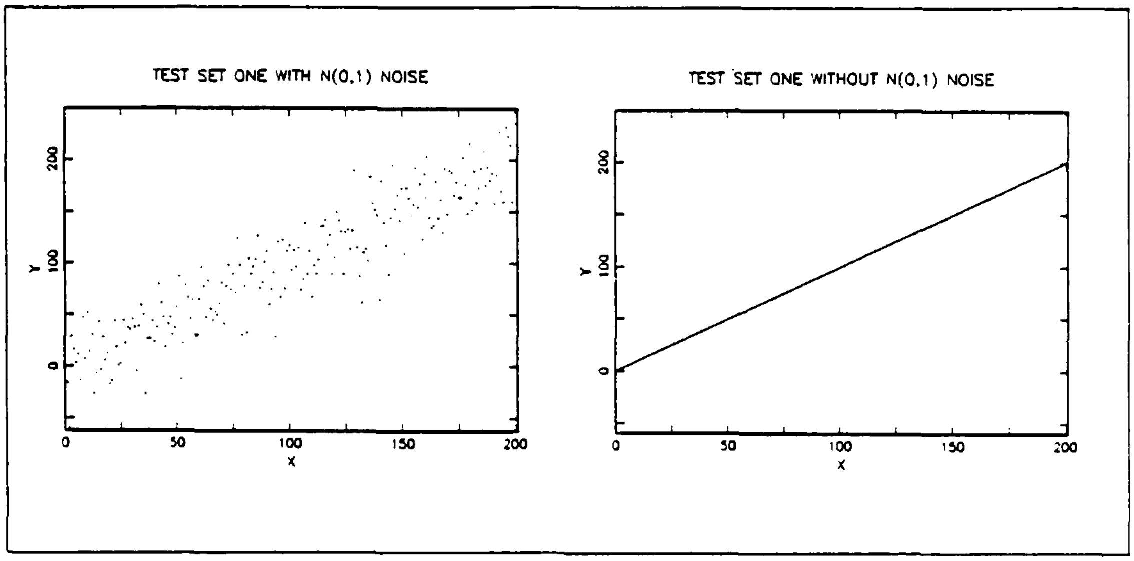


Figure 4.1 Test Set One.

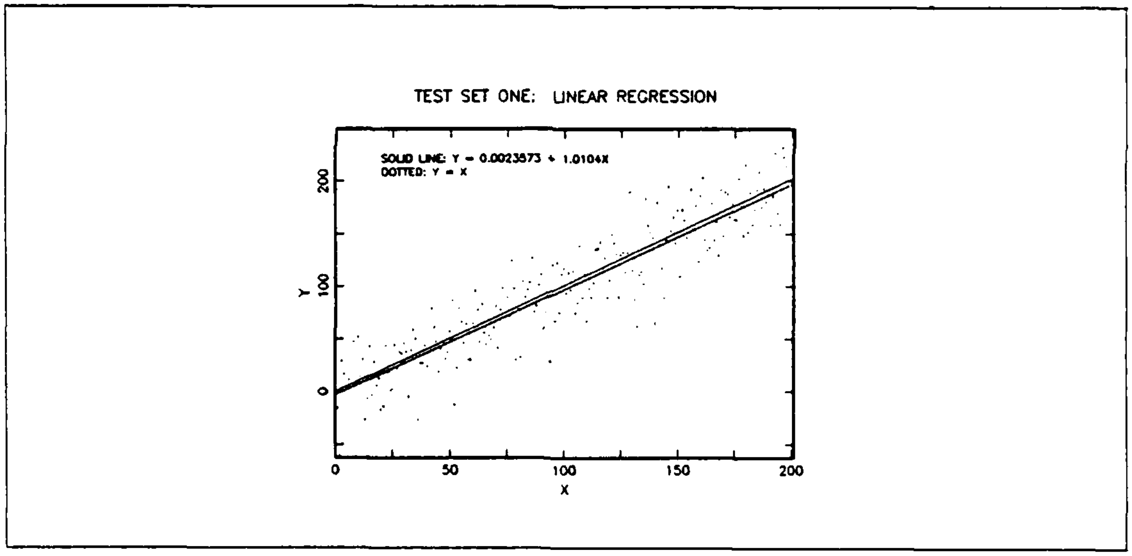


Figure 4.2 Test Set One: Linear Regression.

equal to 0.5 returned the same smooth point values and thus have the same plot.

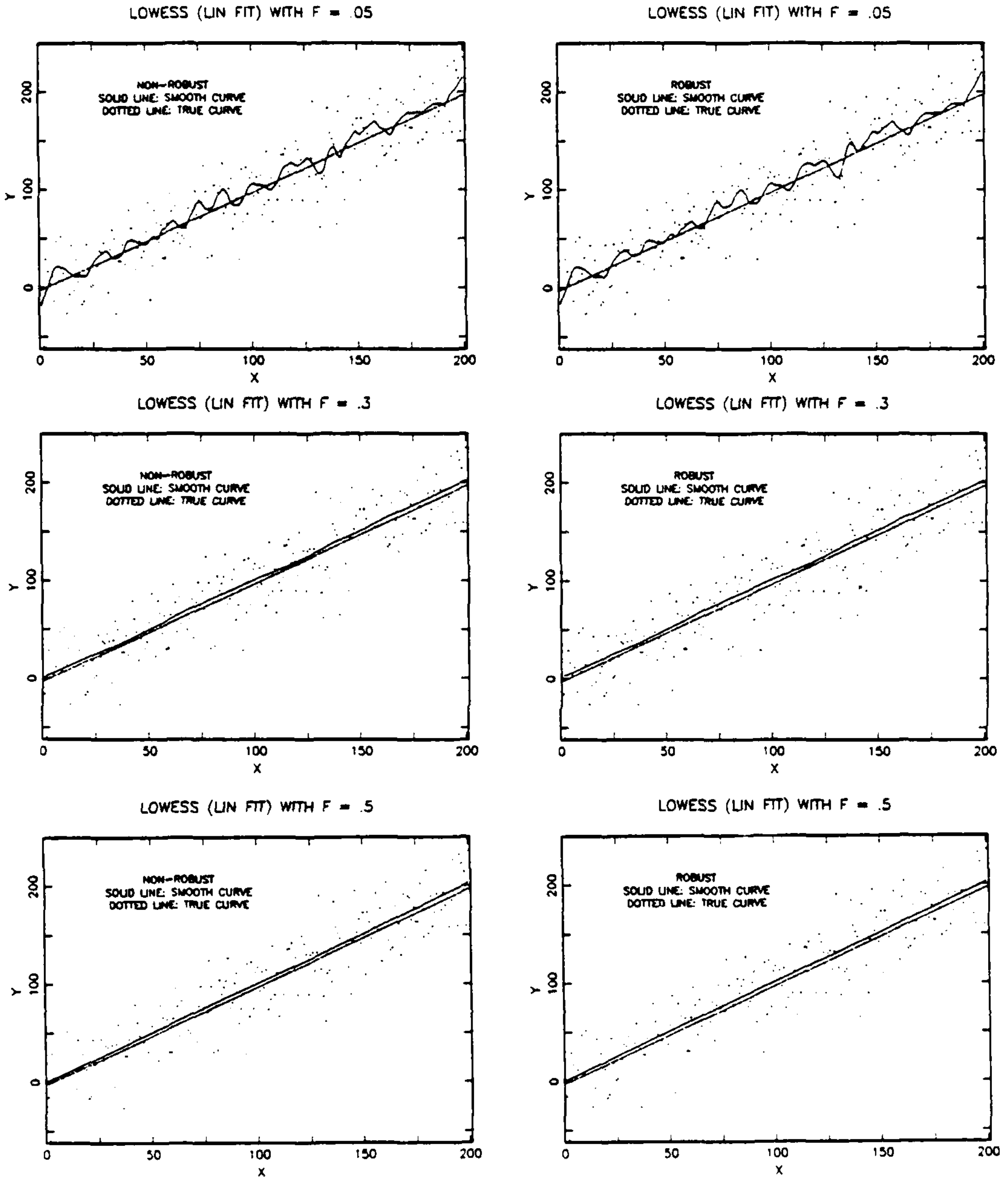


Figure 4.3 Test Set One: LOWESS Smoothing.

Therefore, an individual who has little knowledge of the LOWESS theory should get quick, reasonable results if the F value used is between 0.33 and 0.66 as suggested by Chambers, etal [Ref. 3: p. 98]. The LOWESS smoothing program may have to be run several times before a straight line is produced.

In Figure 4.4 are shown the smoothing results produced by the Supersmoother algorithm. The graphs on the left-hand side display the curves produced when only one span value is used. The graphs on the right are the results of using three span values during each run of Supersmoother. The two top right graphs in Figure 4.4 illustrate the difference between robust smoothing, i.e. ALPHA= 0.0, and non-robust smoothing, i.e. ALPHA= 10.0. The single span value curves in Figure 4.4 are quite similar to the smooth curves produced by LOWESS. The reason for the similarity is that both LOWESS and Supersmoother are central smoothers, i.e. the smooth point value is the result of averaging over the points in the neighborhood.

When three span values are used the smooth poin'.s generated by Supersmoother converge much faster to the underlying linear function than the smooth points generated by LOWESS. Therefore, the Supersmoother algorithm traces very well the linearity of a data set ᵛ ith linear trends.

When the Split Linear Fit algorithm is used with only one window size, the resulting curves, shown on the left-hand side of Figure 4.5, are not much different from the curves produced by LOWESS and Supersmoother. The right-hand graphs of Figure 4.5 illustrate that when the Split Linear Fit algorithm is given more than one window size, the generated smooth point values do not converge to the linear underlying function as fast as Supersmoother. The smallest window size, i.e. 10 and 15, in each case has a great impact on the shape of the smooth curve, because with few points in the window the outliers receive higher weights than in 'large' windows. This illustrates that this smoothing algorithm is designed to place more emphasis on the outlying data points in the data. Equation 1.4 explains that smaller

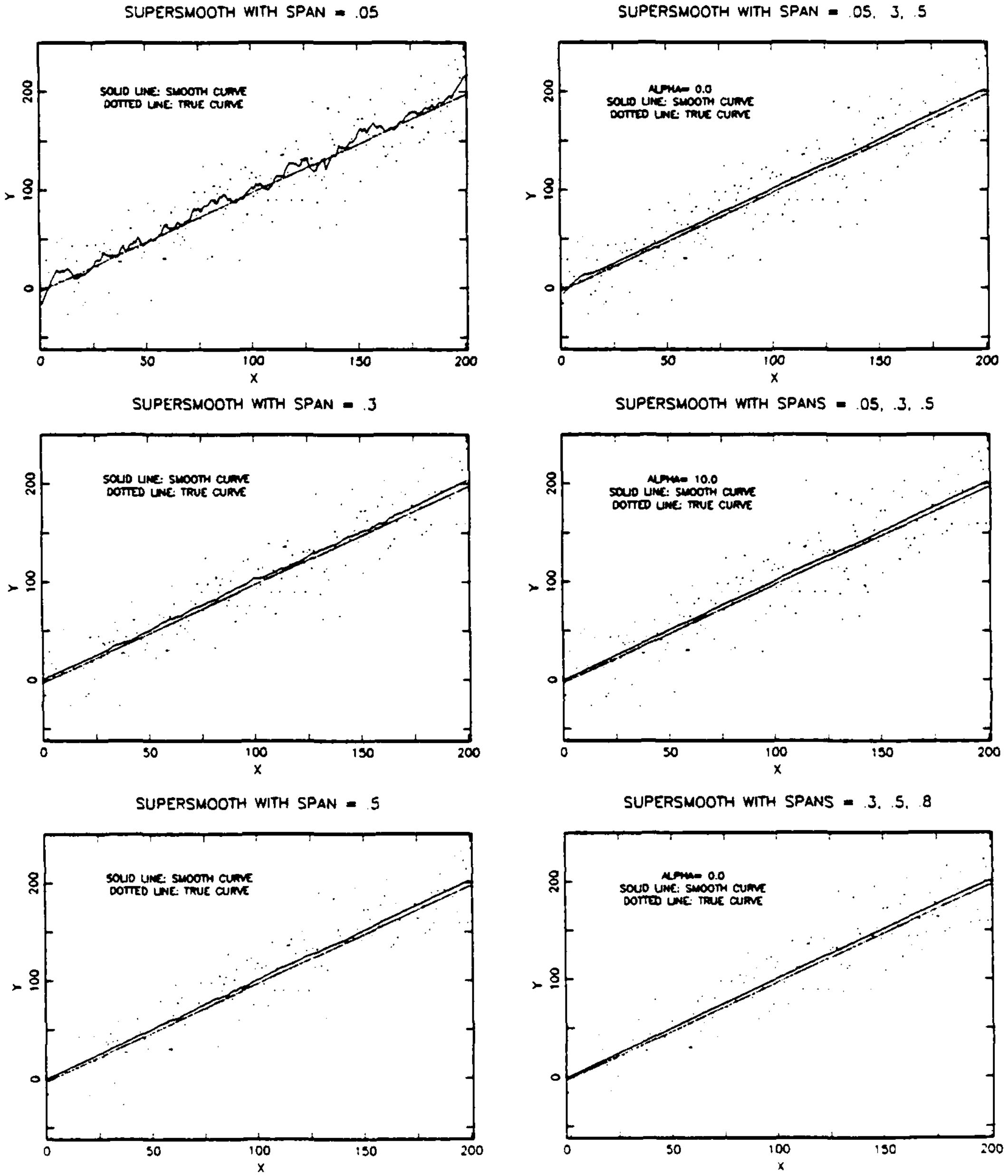


Figure 4.4 Test Set One: Smoothing With Supersmoother.

window sizes will have less bias and thus show more shape, which is the case shown in the right-hand plots of Figure 4.5. Therefore, the user's selection of the smallest window size will determine the degree of convergence toward the linear underlying function.

Figure 4.6 exhibits smoother, more linear curves than were shown in Figure 4.5, the reason being that larger window sizes are used, i.e. increase of bias and decrease of the variance. The bottom graphs of Figure 4.6 demonstrate the effect of increasing the minimum window size, MNWNSZ. In Chapter II of this thesis the use of MNWNSZ is discussed. The author of this thesis, after using the Split Linear Fit for several months, recommends that the size of MNWNSZ be less than one-half the size of the smallest window size being used. The bottom left graph of Figure 4.6 illustrates the distorting effect produced when this recommendation is not followed. The Split Linear Fit algorithm does a good job of depicting data with a linear trend, but the user needs to understand the theory behind the Split Linear Fit, e.g. the window sizes had to be increased in order to produce a smoother curve, in order to achieve acceptable results.

Another measure of performance that can be used in verifying the efficiency of a smoother is the sum of squared residuals. Table 1 shows the sum of squared residuals for the 'best' fitting linear curves produced by each smoother, where 'best' is supported by a plot of the smooth curves shown in this chapter.

All the fits listed in Table 1 produce a fairly straight line which is close to the true underlying function, Y = X. Other fitted curves do produce lower values of sum of squared residuals but the plotted curve deviates from a straight line, e.g. the plot produced by Supersmoother with SPAN(s) = 0.05, 0.3, 0.5 and ALPHA = 0.0 is not very straight, but the sum of squared residuals is 204.6112056. The decrease in the sum of squared residuals is due to an increase of bias. Supersmoother performed almost as well as the Linear Regression and LOWESS, but three neighborhood sizes had to be used, instead of one. The Split Linear Fit smoother did not do as well as Supersmoother for the following reasons:

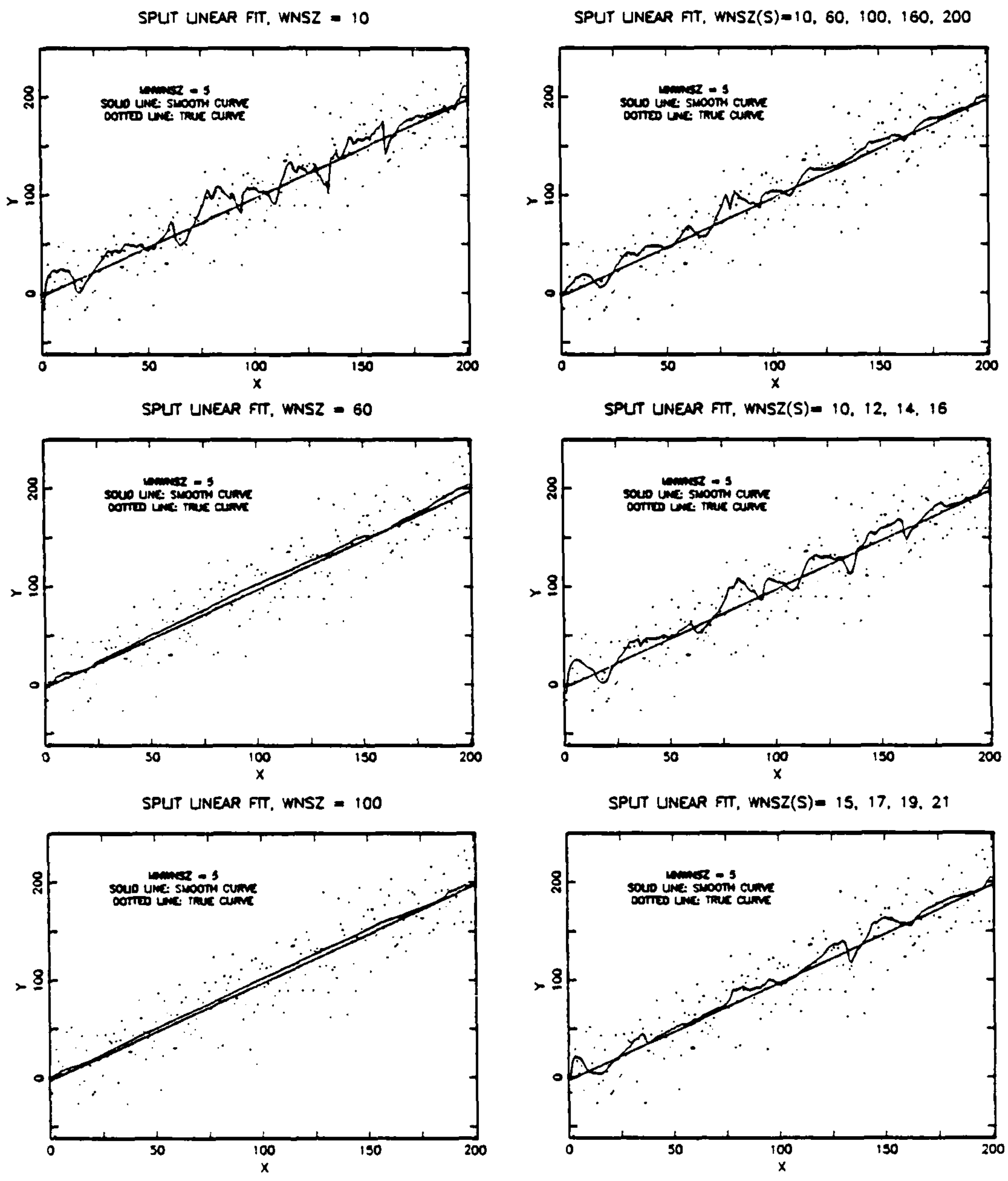


Figure 4.5 Test Set One: Smoothing With Split Linear Fit.

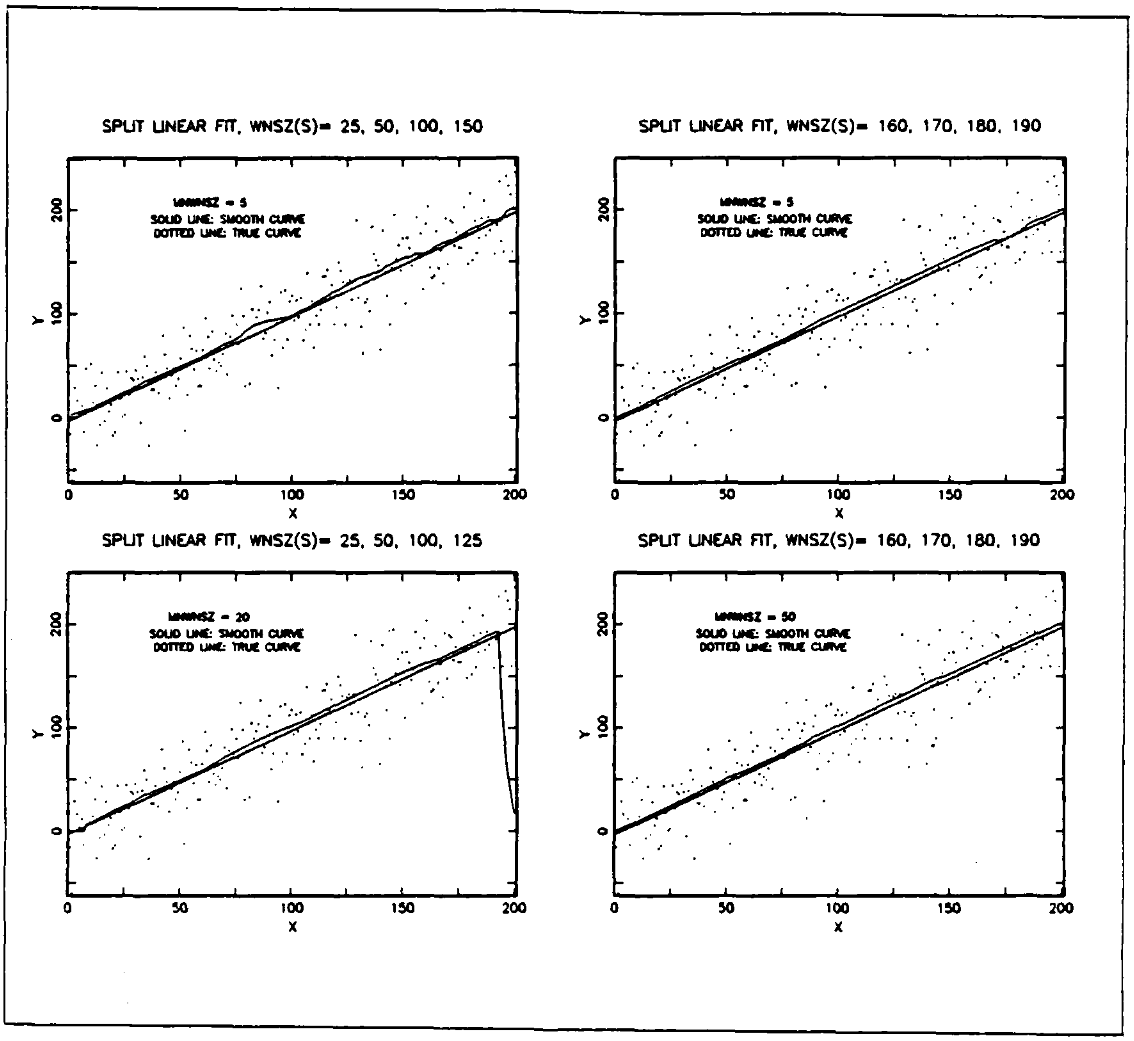


Figure 4.6 Test Set One: Split Linear Fit Change of Smallest Window Size and MNWNSZ.

1. Neighborhood sizes larger than those used by the other smoothers had to be used by Split Linear Fit before a smooth curve could be produced, therefore, this smoother is slower than the other smoothers in converging to a known underlying function.
2. The sum of squared residuals value produced by the best, Split Linear Fit curve is the largest of all the values, therefore, this curve is not as accurate as the other 'best' curves.

TABLE 1

TEST SET ONE:
SUM OF SQUARED RESIDUALS OF THE BEST FITS

Type of Fit	Sum of Squared Residuals
Linear Regression	205.94074
LOWESS, robust, F= 0.5	205.92350
LOWESS, non-robust, F= 0.5	205.82359
Supersmoother, ALPHA= 10.0 SPAN(s)= 0.05, 0.3, 0.5	205.84026
Split Linear Fit, MNWNSZ= 50 WNSZ(s)= 160, 170, 180, 190	206.5657

TABLE 2

TEST SET ONE: COMPUTER CPU CONSUMED

Type of Fit	CPU Consumed (in Seconds)
Linear Regression	1.42
LOWESS, robust, F= 0.5	9.73
LOWESS, non-robust, F= 0.5	3.36
Supersmoother, ALPHA= 10.0 SPAN(s)= 0.05, 0.3, 0.5	1.55
Split Linear Fit, MNWNSZ= 50 WNSZ(s)= 160, 170, 180, 190	2.28

In Table 2 are listed the Central Processing Units, i.e. unit of time used by the IBM 3033 computer in processing a program, consumed when the smoothing techniques listed in Table 1 are used. In order to be consistent in the CPU measurements, each smoother was used to smooth the same data set and place the smoothed output in an APL variable. The CPU times listed in the table indicate that the advanced smoothers do better than most of the other smoothers, but the improvement is only in seconds. Therefore, a user trying to select between the smoothers should balance this saving in CPU time with the cost of computer and personnel time, before deriving a conclusion.

D. TESTING AND RESULTS----SMOOTH CURVATURE IN UNDERLYING FUNCTION

The second test is designed to test how the Supersmoother and the Split Linear Fit algorithms perform on a data set which has an underlying function with smooth curvature, i.e. the change from one point to the next is not abrupt. Figure 4.7 displays Test Set Two which consists of 200 data points generated with the following equation:

$$Y = \mathrm{COS}\left(\frac{X}{25}\right) + \text{Normal (0,1) noise},\ 0<X\leqslant 200\ . \tag{4.4}$$

The values generated by this function and used in this section are in tabular form in Appendix D.

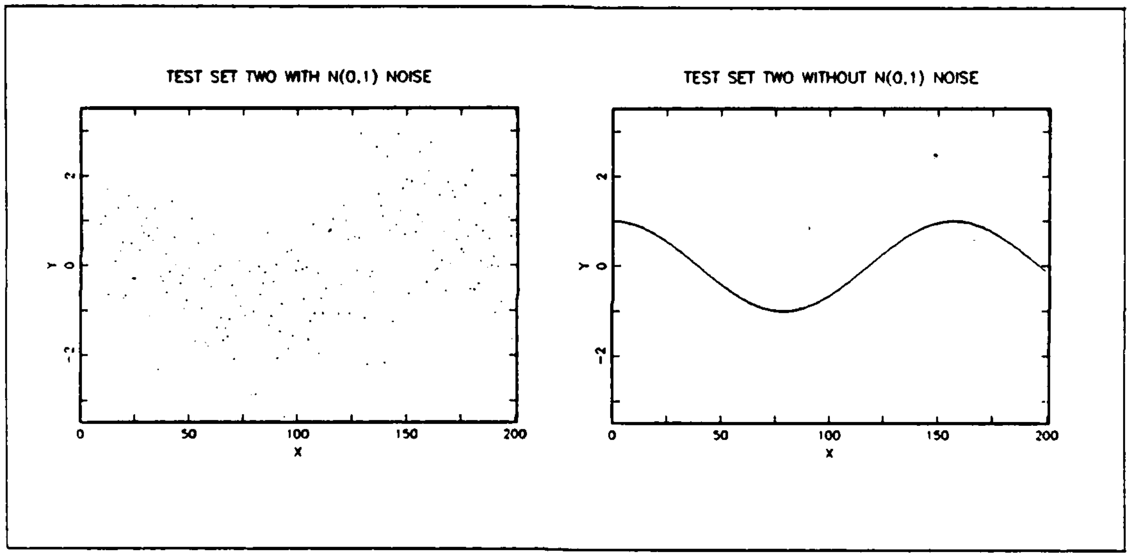


Figure 4.7 Test Set Two.

Figure 4.8 shows the curve fitting results of third degree polynomial curve fit to Test Set Two. Confidence Interval Tests on the coefficients of the equation shown on Figure 4.8 reveal that the coefficients are not significantly different from zero with a Confidence Level of less than 0.001, thus the coefficients should be accepted.

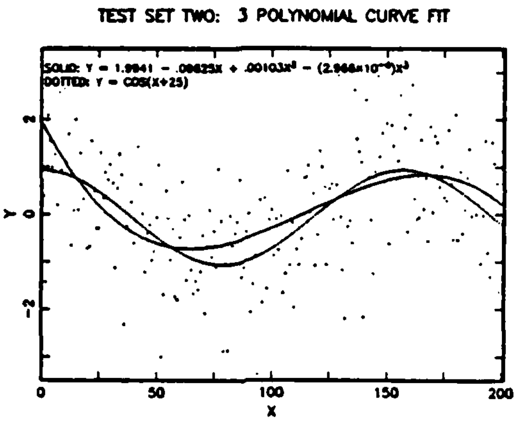


Figure 4.8 Test Set Two: Third Degree Polynomial Curve Fit.

Figure 4.9 shows the results of the Spline fit, as previously discussed this GRAFSTAT function requires that the user enter the maximum sum of squared residuals value. In Figure 4.9 0 indicates that the smooth function should have a sum of squared residuals less than or approximately equal to the second parameter.

Figure 4.10 shows the smooth curves produced from using the Equal-Weight Moving Average smoother. As usual a larger neighborhood size, M, results in a smoother curve. As stated in Chapter I, the disadvantage of using the Equal-Weight Moving Average smoother is that smooth data points are dropped from the ends of the output data set. This is illustrated in Figure 4.10 where with M=60, 30 points are dropped from each end.

Figure 4.11 shows the results produced by the Cosine-Weighted Moving Average smoother. Since this smoother is an extension of the Equal-Weight Moving Average, it can be seen that as the neighborhood size, M, is increased smooth point values are also dropped from the ends of the data set. In addition, this figure illustrates how the smooth curve converges toward the true underlying function and then

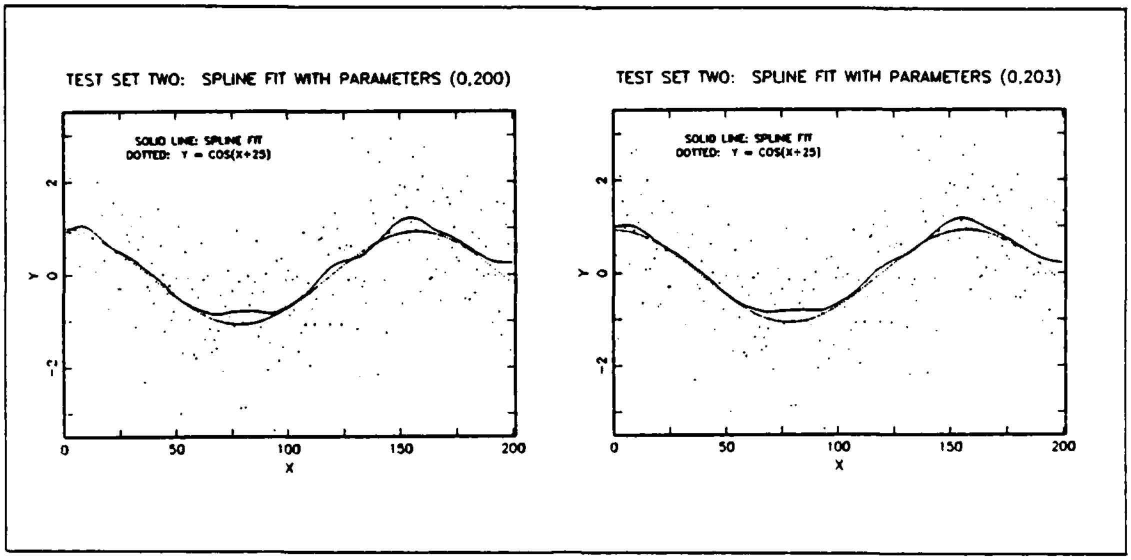


Figure 4.9 Test Set Two: Spline Fit.

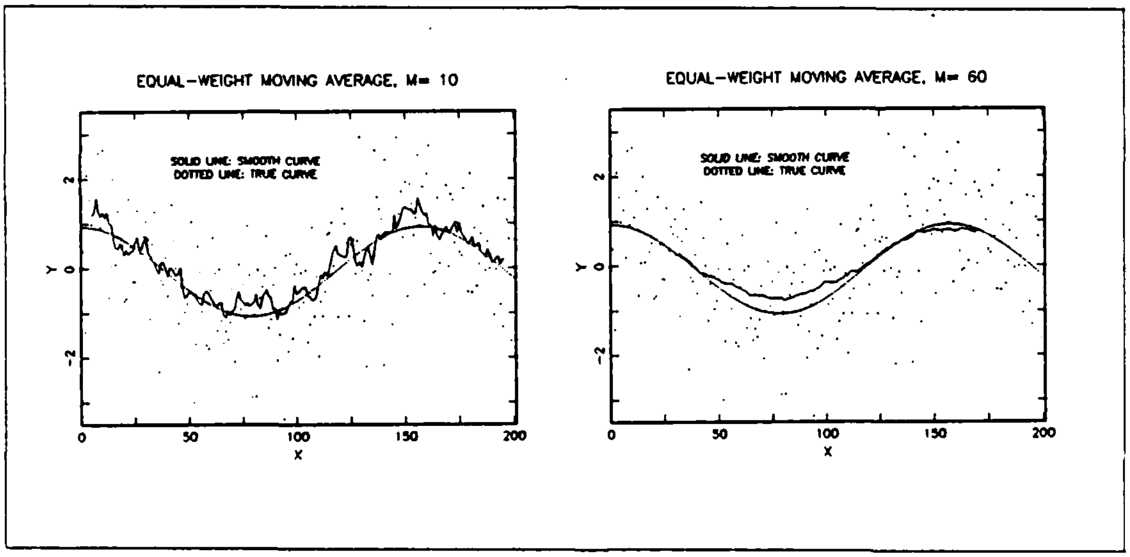


Figure 4.10 Test Set Two:
Equal-Weight Moving Average Smoothing.

away from this same underlying function, i.e. the variance decreases but bias increases beyond a certain point.

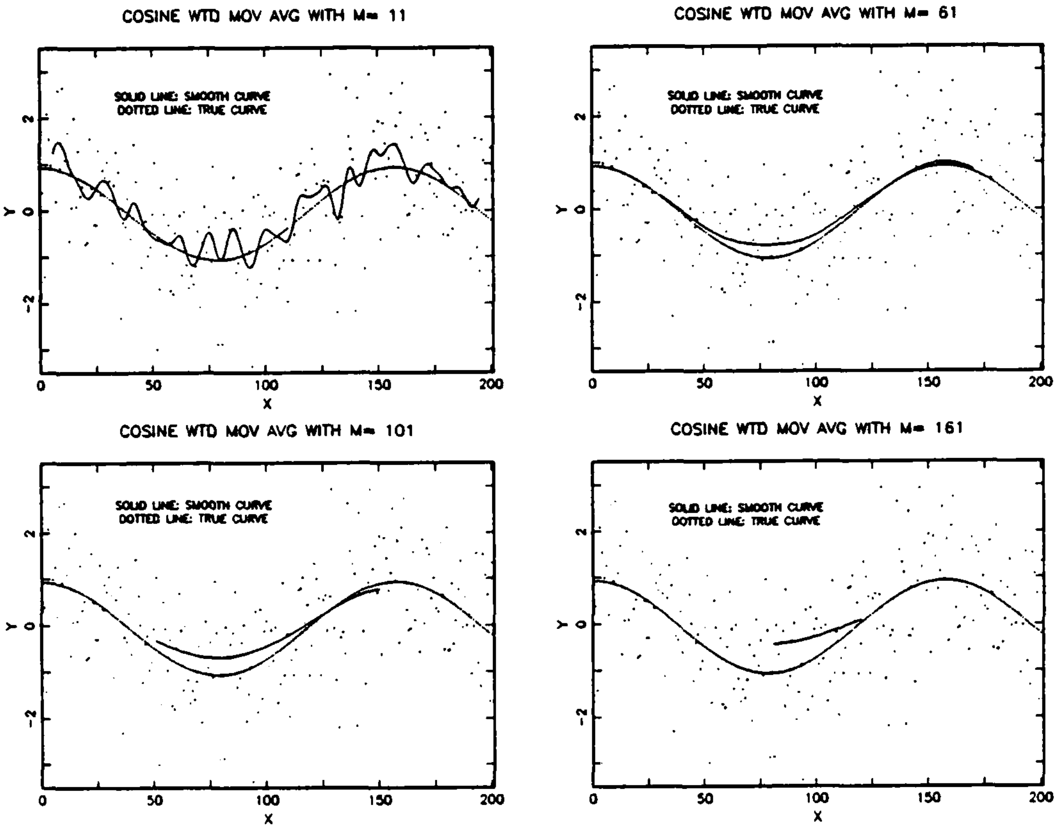


Figure 4.11 . Test Set Two:
Cosine-Weighted Moving Average Smoothing.

The LOWESS smooth results are shown in Figure 4.12. Since Test Set Two appears to be non-linear, the quadratic fitting option of LOWESS is used. Only the robust cases are shown since the results are basically the same as the non-robust cases. The best fitting curve appears to be the smooth curve produced by using F= 0.5.

The Supersmoother results are displayed in Figure 4.13. The smooth curves produced by using three span values tend to maintain the shape of the underlying function across most span values.

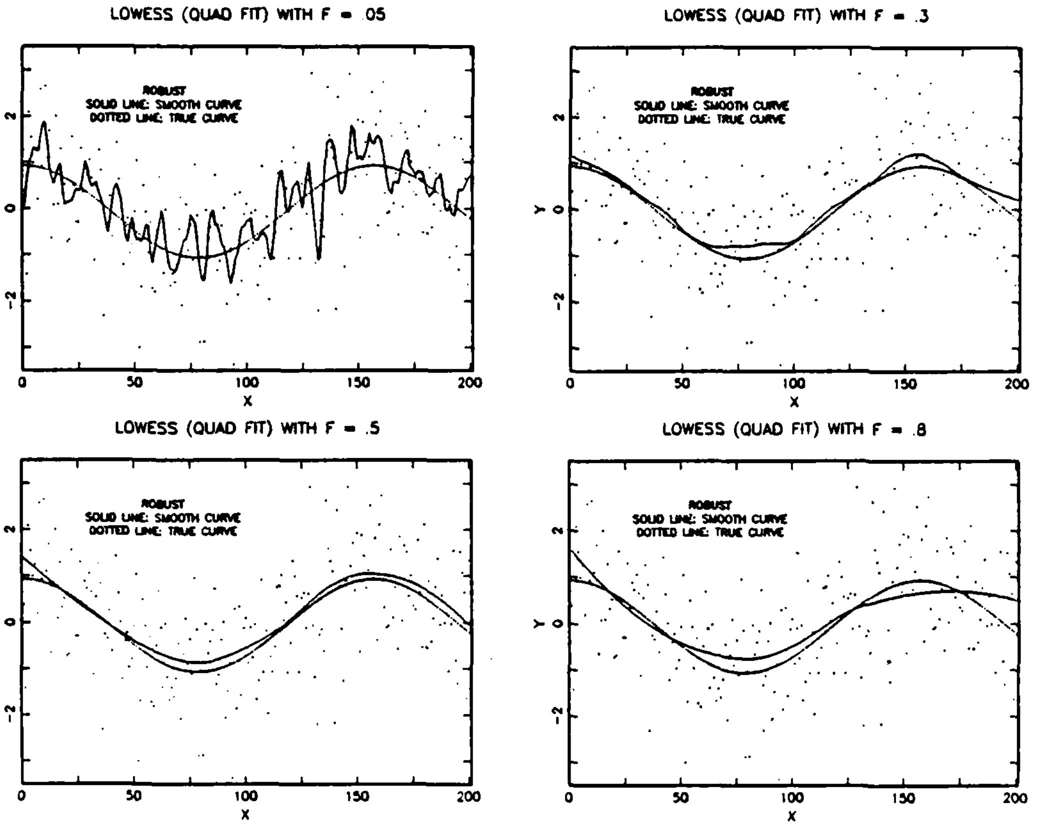


Figure 4.12 Test Set Two: LOWESS Smoothing.

However, if only one span value is used, the shape of the smooth curve approaches a straight line as the size of the span value approaches the value of 1.0. The top right and the middle right graphs illustrate the difference between using robust smoothing, i.e. ALPHA= 0.0, and non-robust smoothing, i.e. ALPHA= 10.0. Therefore, it is best to use Supersmoother with three span values and ALPHA=0.0.

Figure 4.14 displays a radical difference between using a single window size and several window sizes as input to the Split Linear Fit algorithm. As stated in Chapter III, the main purpose of the Split

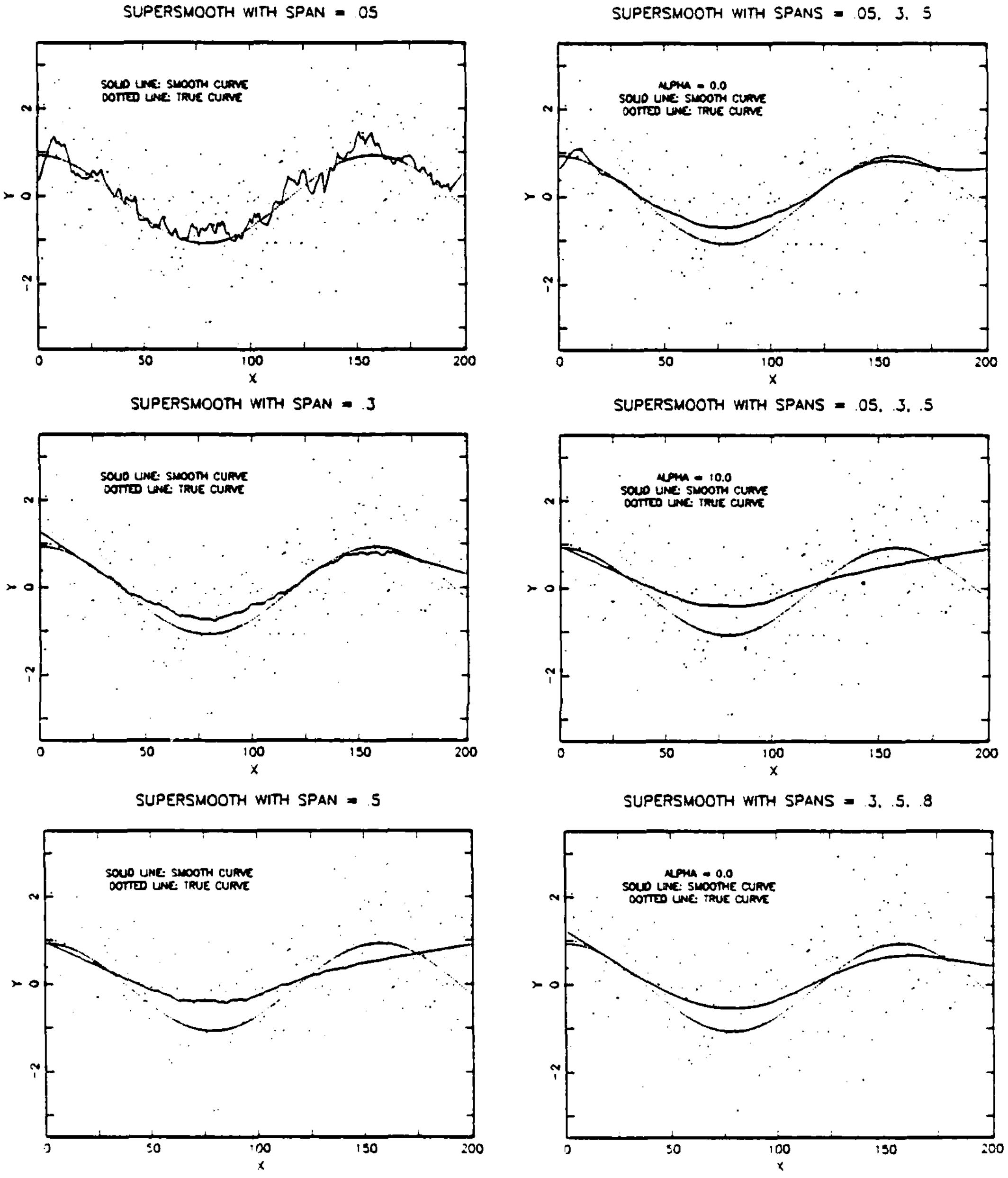


Figure 4.13 Test Set Two: Smoothing With Supersmoother.

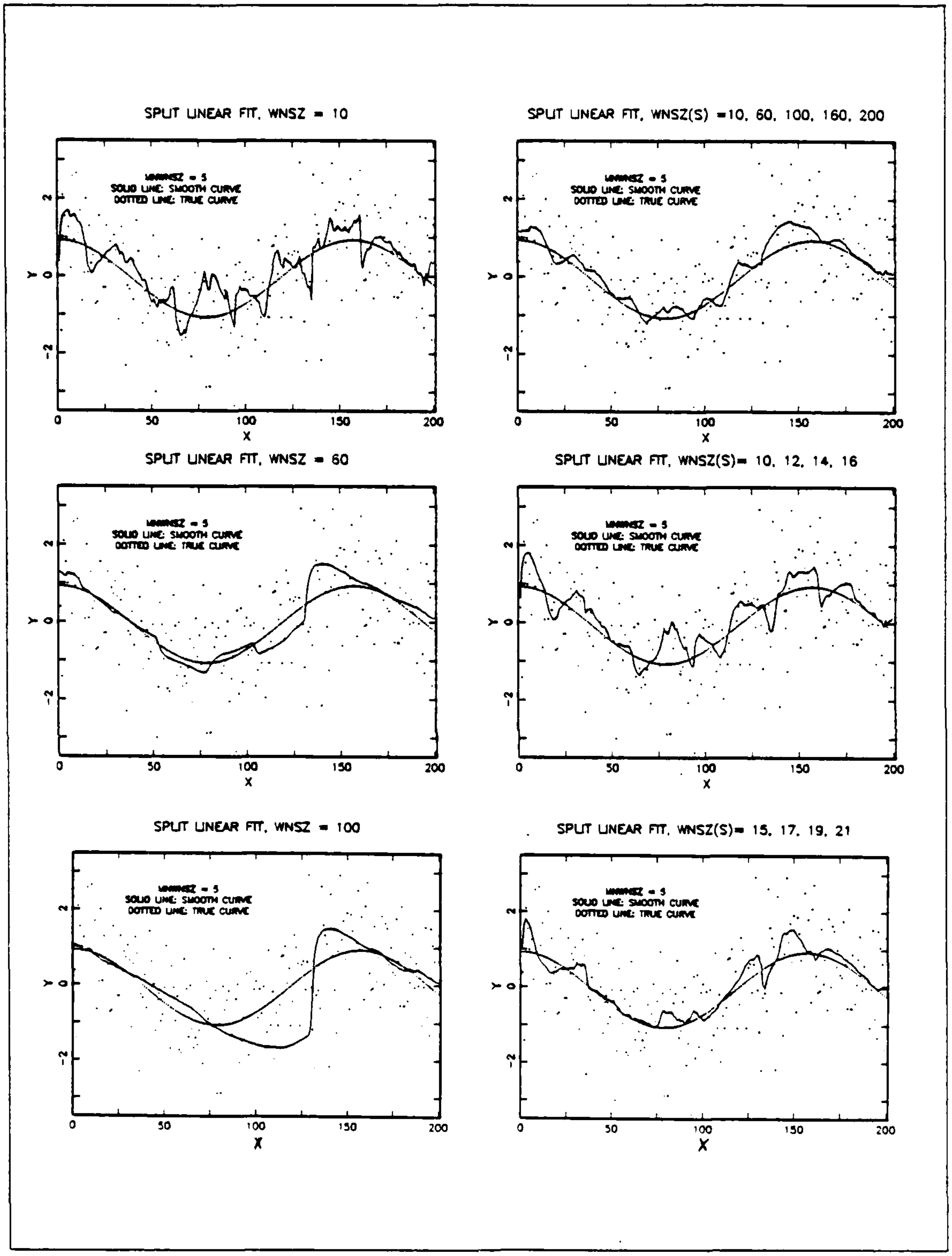


Figure 4.14 Test Set Two: Smoothing With Split Linear Fit.

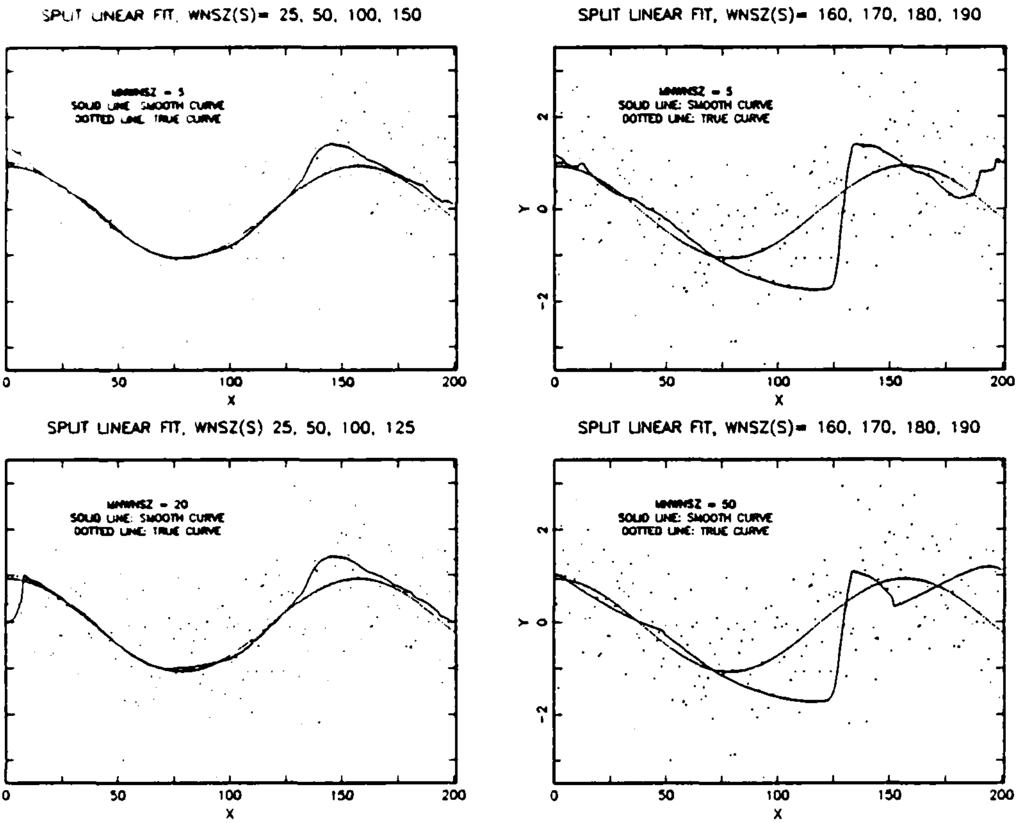


Figure 4.15 Test Set Two: Split Linear Fit Change of MNWNSZ.

Linear Fit smoother is to disclose sudden abrupt changes in a data set. By using a single window size, the algorithm does not produce enough information about the true shape of the given data set, and the result is the deviating smooth curves on the left side of Figure 4.14. The use of several window sizes produces more information about the raw data, and the result is smoother curves; see the graphs on the right of Figure 4.14. Figure 4.15 shows the effect produced when the size of MNWNSZ is changed and when large window are used. The smoothed output produced by the Split Linear Fit smoother never converges to the true underlying function no matter what window sizes are used.

The sum of squared residuals corresponding to the 'best' fitting curves are listed in Table 3. Each of the fits listed in Table 3 has a different degree of convergence toward the true underlying function, Y = COS(X/25). Other fits not listed in Table 3 but discussed in this section produced smaller sum of squared residuals, however the smooth curve did not resemble the true underlying function to a high degree. Supersmoother performs fairly well when given a set of data with a smooth curvature. The Split Linear Fit smoother did a poor tracing the true underlying curve very well, when compared to other more simple smoothers.

TABLE 3

TEST SET TWO:
SUM OF SQUARED RESIDUALS OF THE BEST FITS

Type of Fit	Sum of Squared Residuals
Third Degree Polynomial Curve Fit	219.14846
Spline Fit (0, 203)	203.70213
Equal-Weight Moving Average, M= 60	159.04903
Cosine-Weighted Moving Average, M= 61	158.38194
LOWESS, robust, F= 0.5	207.26
LOWESS, non-robust, F= 0.5	207.133
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.05, 0.3, 0.5	209.28674
Split Linear Fit, MNWNSZ= 5 WNSZ(s)= 25, 50, 100, 150	209.51462

Table 4 shows the CPU times consumed by the smoothers listed in Table 3. The Cosine-Weighted Moving Average smoother, in addition to producing a very good sum of squared residuals value, is very fast in generating the smoothed results. The advanced smoothers were much slower than the Cosine-Weighted Moving Average, but were faster than LOWESS.

TABLE 4

TEST SET TWO: COMPUTER CPU CONSUMED

Type of Fit	CPU Consumed (in Seconds)
Third Degree Polynomial Curve Fit	0.96
Spline Fit (0, 203)	12.15
Equal-Weight Moving Average, M= 60	0.06
Cosine-Weighted Moving Average, M= 61	0.22
LOWESS, robust, F= 0.5	21.64
LOWESS, non-robust, F= 0.5	7.13
Supersmoother, ALPHA= 10.0 SPAN(s)= 0.05, 0.3, 0.5	1.55
Split Linear Fit, MNWNSZ= 50 WNSZ(s)= 160, 170, 180, 190	2.37

E. TESTING AND RESULTS----ABRUPT CHANGES IN CURVATURE IN UNDERLYING FUNCTION

The third test examines the performance of the Supersmoother and the Split Linear Fit algorithms on a data set which includes a triangular function. Test Set Three is shown in Figure 4.16 and the point values are displayed in table form in Appendix D. The following equation was used to generate Test Set Three:

$$Y = \begin{cases} 1.0 - 0.06X & \text{, if } 0<X\leqslant 50 \\ 8.0 - 0.08X & \text{, if } 50<X\leqslant 100 \\ -0.808 + 0.008X & \text{, if } 100<X\leqslant 150 \\ 0.392 & \text{, if } 150<X\leqslant 200\,. \end{cases} \qquad (4.5)$$

In order to check the data set against equation 4.5 a linear regression was done on each part of the above equation and the results are displayed in Figure 4.17. This figure shows that the regression equations deviate from the true equations. A Confidence Interval test done on each of the coefficients indicates that the coefficients are significantly different from zero, therefore, the equations produced by the regression are accepted.

The curve resulting from third degree polynomial fit on Test Set Three is shown in Figure 4.18. A Confidence Interval test done on the coefficients reveals that the coefficients produced by this fitting technique are not significantly different from zero with a Confidence Level less than 0.001. Thus all the coefficients in the equation shown in Figure 4.18 are accepted.

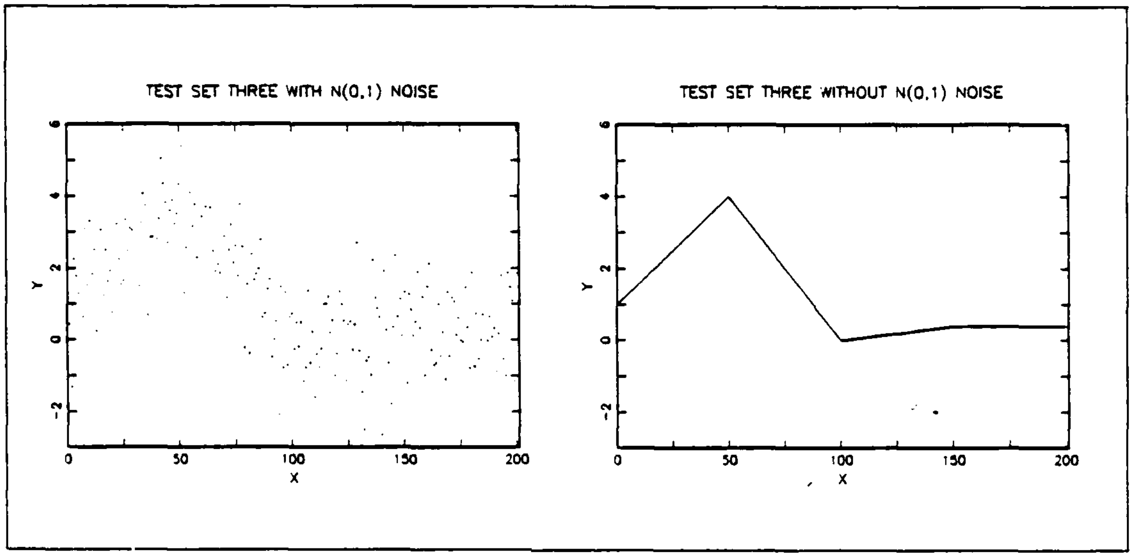


Figure 4.16 Test Set Three.

A plot of the point values generated by a Spline Fit with the sum of squared residuals required to be no greater than 204 is shown in Figure 4.19.

The smooth point values generated on two runs of the Equal-Weight Moving Average are plotted, and the curves are displayed in Figure 4.20.

The curves produced by plotting the smooth point values generated by the Cosine-Weighted Moving Average smoother are displayed in Figure 4.21.

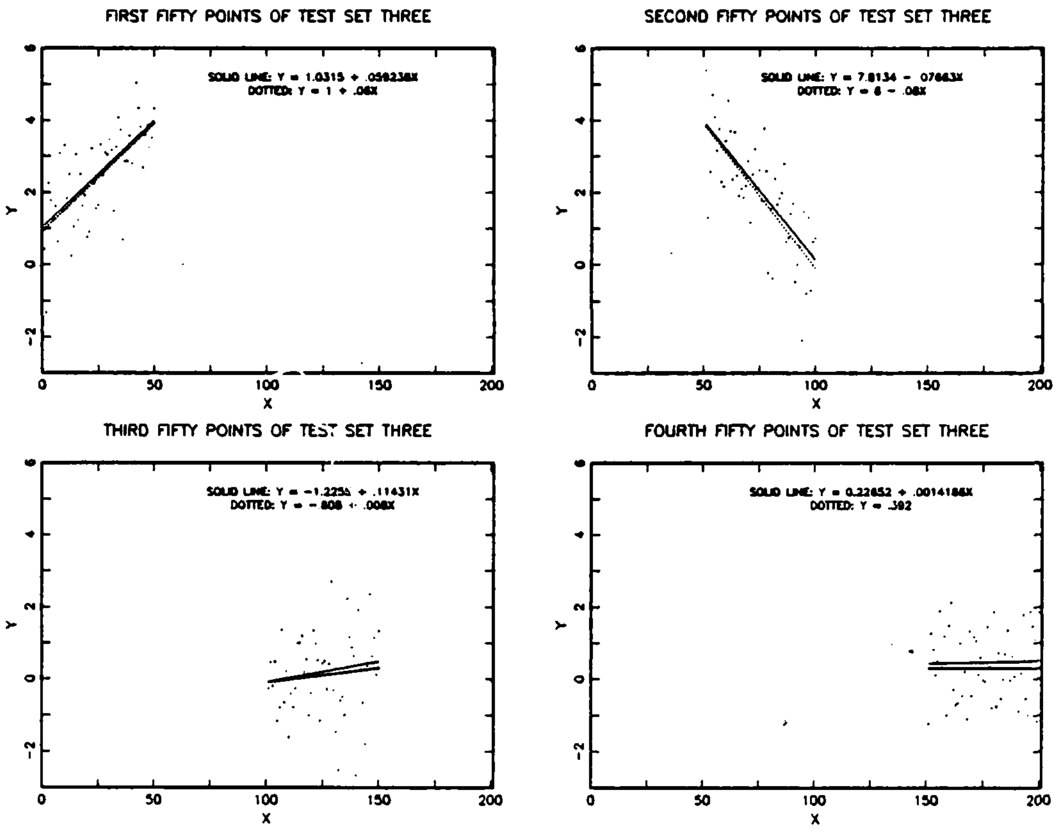


Figure 4.17 Test Set Three Broken Into Four Linear Sections.

The LOWESS smoothing results are displayed in Figure 4.22. The best fit is produced by the run with F= 0.3. The smooth curve almost coincides with the true underlying function.

Figure 4.23 contains the plots of the smooth points produced by Supersmoother. As can be seen, the best fit is when three span values are used in the smoothing, i.e. SPAN(s)= 0.05, 0.3, 0.5, which happen to be starting values recommended by the Friedman and Stuetzle [Ref. 9: p. 9]. The smooth curve tends to depict the true underlying function very well. The top right graph and the middle right graph

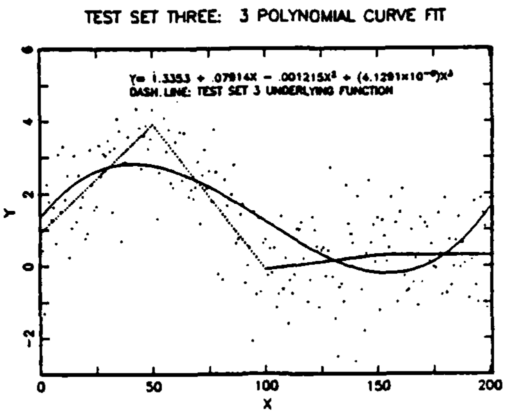


Figure 4.18 Test Set Three: Third Degree Polynomial Curve Fit.

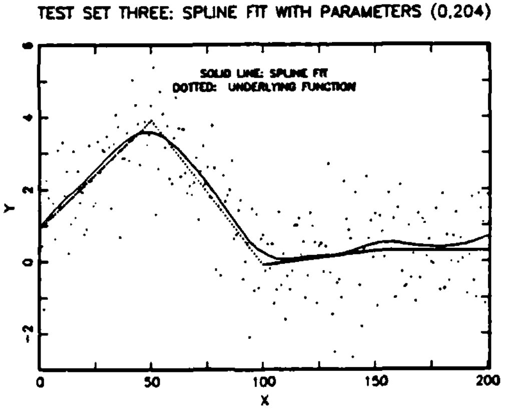


Figure 4.19 Test Set Three: Spline Fit.

show the great disparity between using non-robust smoothing, i.e. ALPHA= 10.0, and robust smoothing, i.e ALPHA= 0.0. Thus with the Supersmoother three span values with a ALPHA=0.0 are necessary to get useful results.

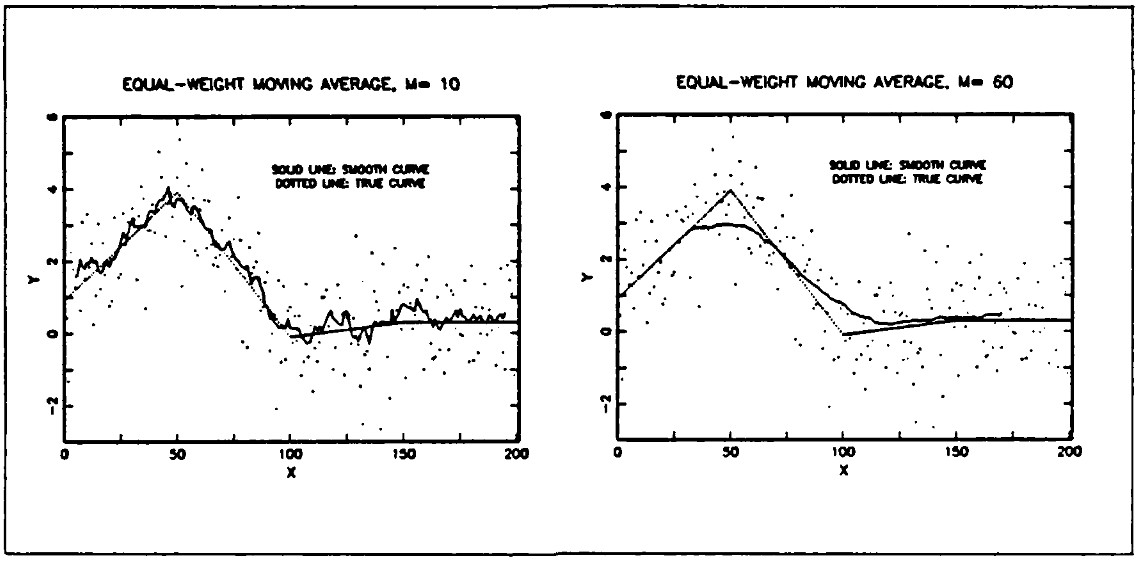


Figure 4.20 Test Set Three:
Equal-Weight Moving Average Smoothing.

By examining the plots in Figure 4.24 and 4.25, it is obvious that the Split Linear Fit smoother produces results which are often too erratic and not useful. Outlying points have too much influence on the output produced by this smoother. The only plot without any drastic deviations from the curvature of the underlying function is the top left plot in Figure 4.25.

Table 5 shows for the third time that in addition to not producing good smooth curves which depict the underlying function very well, the advanced smoothers do not produce sum of squared residuals values as good as the baseline smoothers.

Table 6 shows that the advanced are consistently using the same low amount of CPU time. The LOWESS has fluctuated in CPU usage, but has always used the most CPU. The Cosine-Weighted Moving Average smoother for the second time has generated a very good sum of squared residuals value and has used the least amount of CPU.

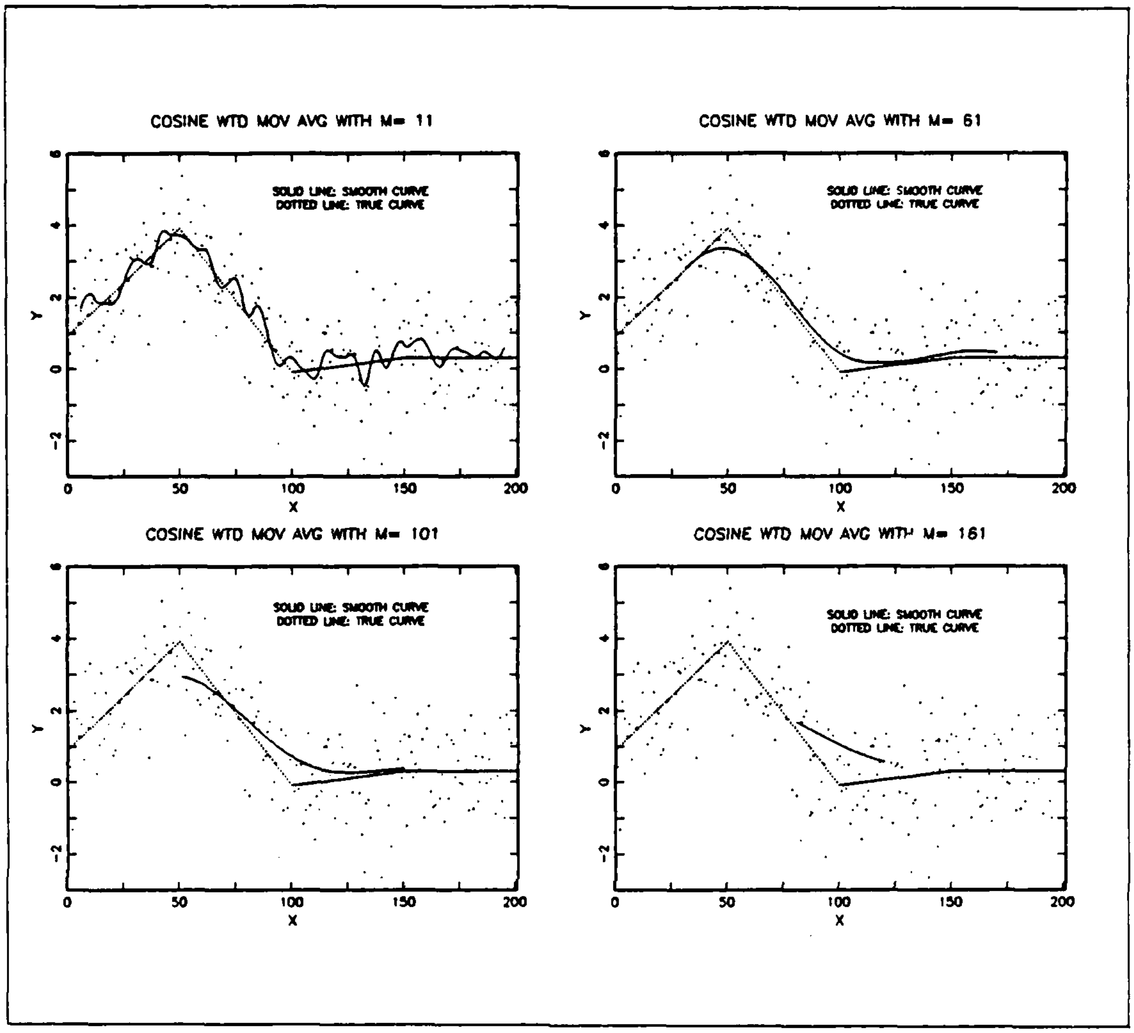


Figure 4.21 Test Set Three:
Cosine-Weighted Moving Average.

F. CONCLUSIONS

The advanced smoothers, Supersmoother and Split Linear Fit, are quite complex to thoroughly understand and require that the user enter many parameters. The interrelationship between the parameters is not clear, and the results are difficult to control. For example, the smallest neighborhood size used by the advanced smoothers has more influence on the output than the other neighborhood sizes, but this value can not be too big or the output will get distorted. In addition,

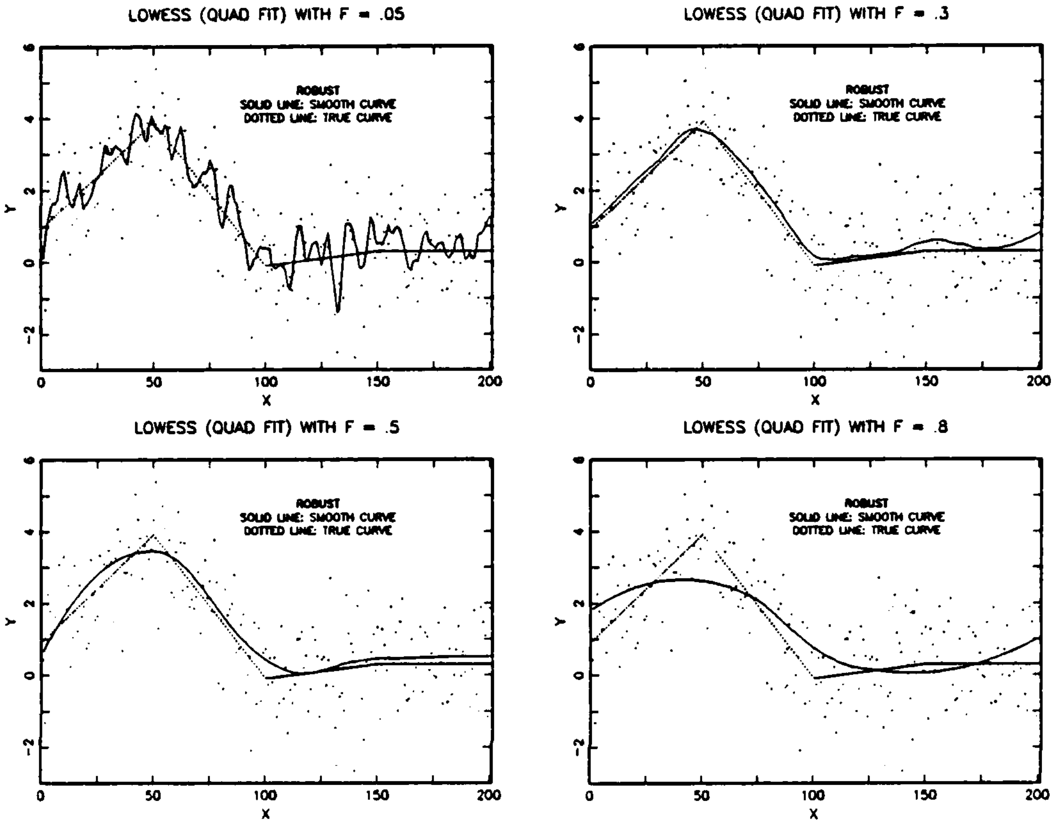
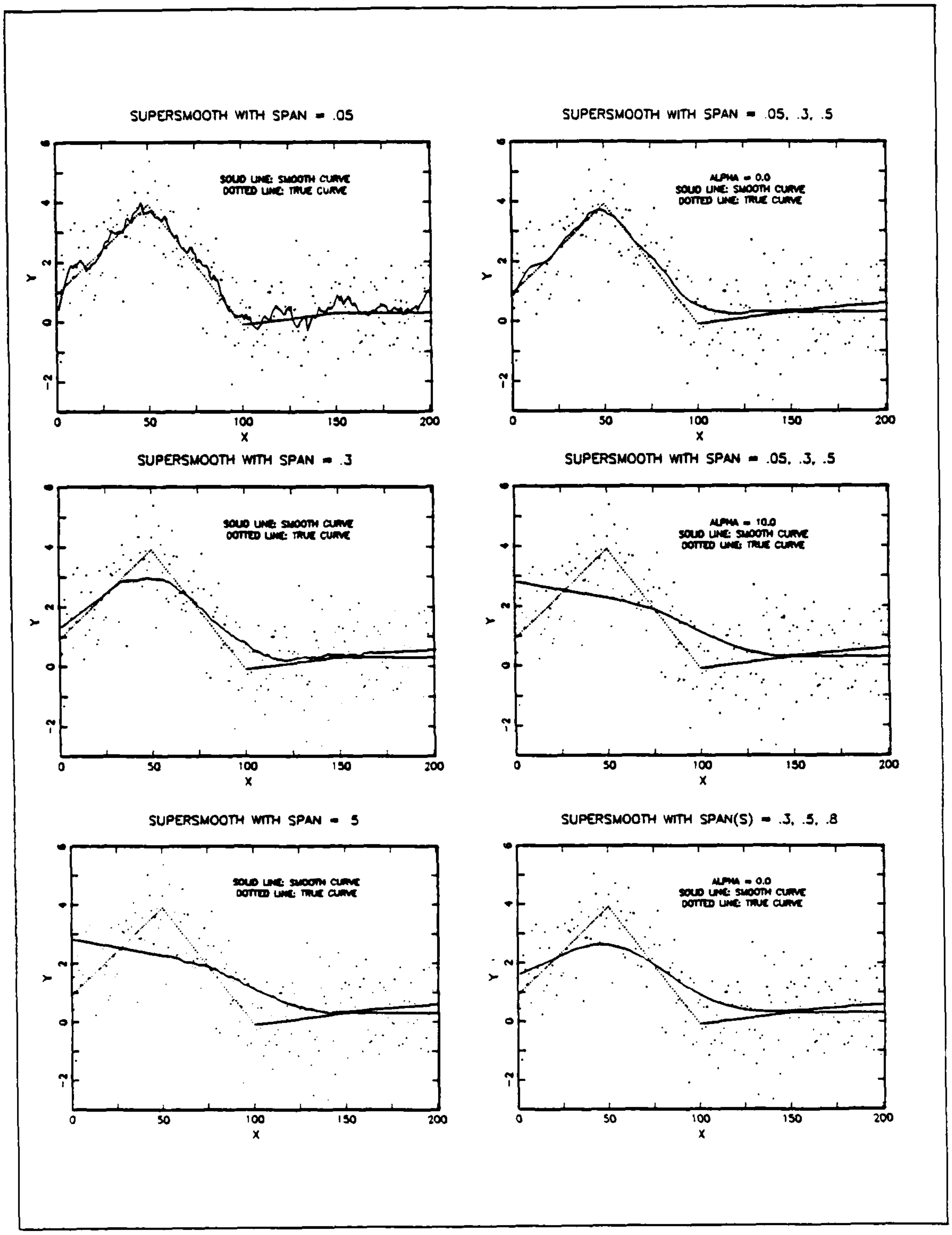


Figure 4.22 Test Set Three: LOWESS Smoothing.

to the three or more neighborhood sizes that the user must think about and calculate, a fourth value must be considered, i.e. the Supersmoother's ALPHA value and the Split Linear Fit's MNWNSZ value. Changing anyone of these values in the advanced smoothers produces radical changes in the shape of the fitted curve, leaving the user confused as to which values to change and by how much. Friedman and Stuetzle recommend that the neighborhood sizes be between 5 and 50 percent of N, where N is the number of points to be smoothed. They also claim that "savings are substantial" [Ref. 9: p. 5], i.e. in the



SUPERSMOOTH WITH SPAN = .05

SUPERSMOOTH WITH SPAN = .05, .3, .5

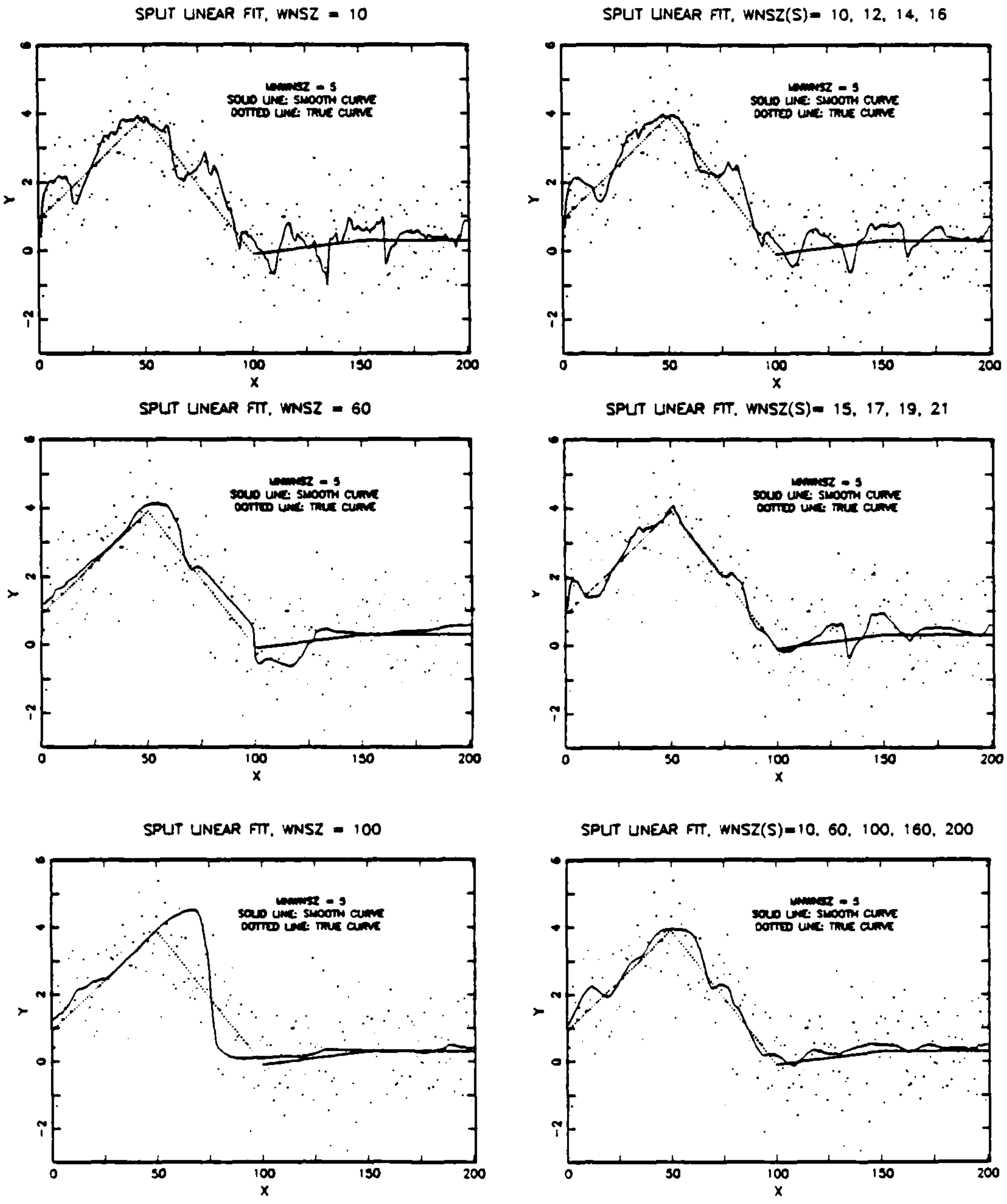
SUPERSMOOTH WITH SPAN = .3

SUPERSMOOTH WITH SPAN = .05, .3, .5

SUPERSMOOTH WITH SPAN = 5

SUPERSMOOTH WITH SPAN(S) = .3, .5, .8

Figure 4.23 Test Set Three:
Smoothing With Supersmoother.



SPLIT LINEAR FIT, WNSZ = 10

SPLIT LINEAR FIT, WNSZ(S)= 10, 12, 14, 16

SPLIT LINEAR FIT, WNSZ = 60

SPLIT LINEAR FIT, WNSZ(S)= 15, 17, 19, 21

SPLIT LINEAR FIT, WNSZ = 100

SPLIT LINEAR FIT, WNSZ(S)=10, 60, 100, 160, 200

Figure 4.24 Test Set Three:
Smoothing With Split Linear Fit.

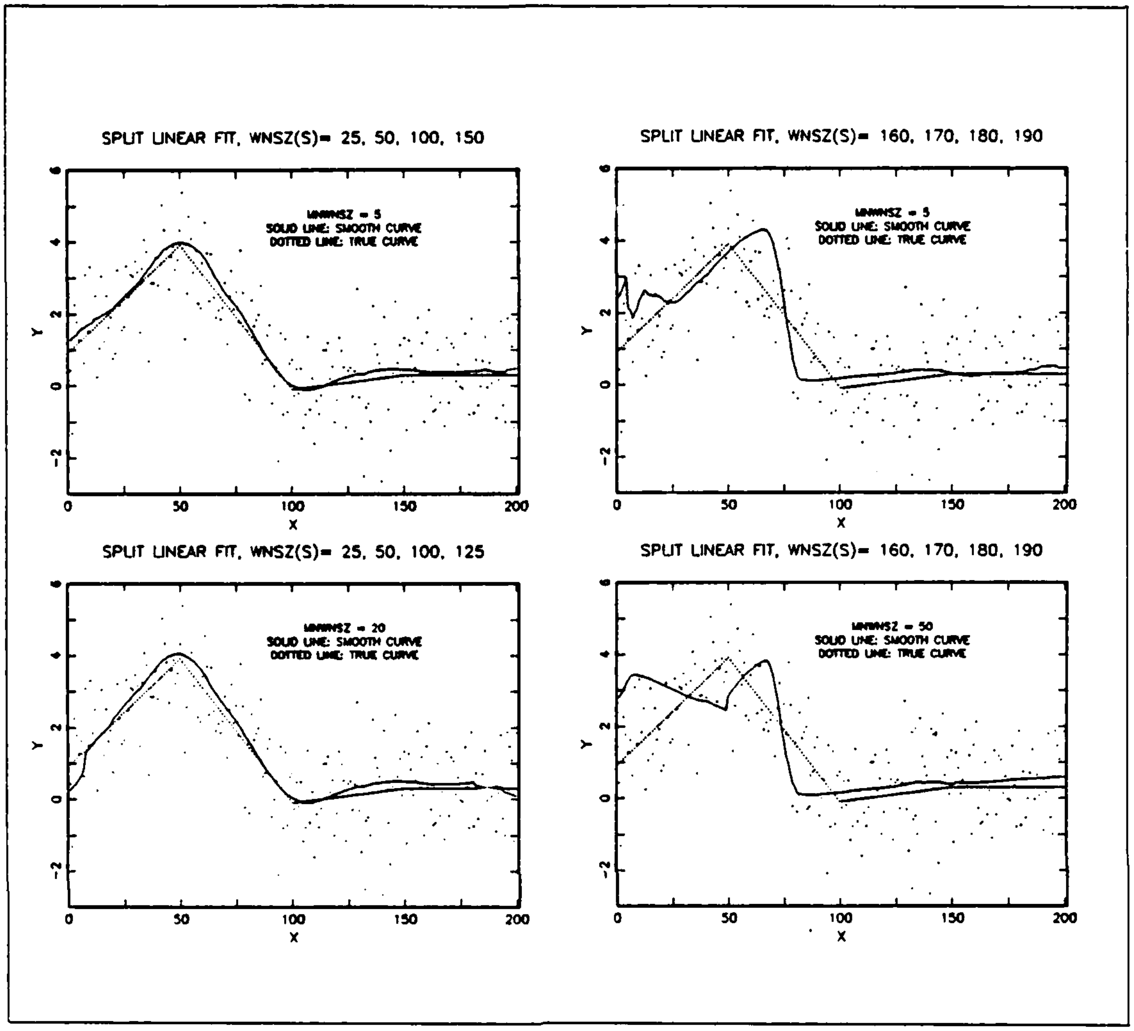


Figure 4.25 Test Set Three: Split Linear Fit, Change of MNWNSZ.

time required to find a desired smoothed curve is greatly reduced. The SPAN values used in this thesis meet this criteria. The program runs fast, but the sum of squared residuals values produced are not as good as the values produced by the simpler, more user friendly smoothers, one of which uses far less CPU.

McDonald and Owen never really give any guidance on the number of window sizes to use except that they used "several (typically three to five)" [Ref. 10: p. 2] in their testing of the Split Linear Fit

TABLE 5

TEST SET THREE:
SUM OF SQUARED RESIDUALS OF THE BEST FITS

Type of Fit	Sum of Squared Residuals
Third Degree Polynomial Curve Fit	270.05338
Spline Fit (0, 204)	204.66025
Equal-Weight Moving Average, M= 60	173.46090
Cosine-Weighted Moving Average, M= 61	156.80284
LOWESS, robust, F= 0.3	204.173
LOWESS, non-robust, F= 0.3	210.571
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.05, 0.3, 0.5	204.70795
Split Linear Fit, MNWNSZ= 5 WNSZ(s)= 25, 50, 100, 150	206.59216

TABLE 6

TEST SET THREE: COMPUTER CPU CONSUMED

Type of Fit	CPU Consumed (in Seconds)
Third Degree Polynomial Curve Fit	0.98
Spline Fit (0, 203)	12.95
Equal-Weight Moving Average, M= 60	0.07
Cosine-Weighted Moving Average, M= 61	0.22
LOWESS, robust, F= 0.5	15.19
LOWESS, non-robust, F= 0.5	5.02
Supersmoother, ALPHA= 10.0 SPAN(s)= 0.05, 0.3, 0.5	1.53
Split Linear Fit, MNWNSZ= 50 WNSZ(s)= 160, 170, 180, 190	2.37

smoother. The smooth curves produced by this smoother never are consistent, i.e. follow a pattern which can be used as a guide toward the desired smooth curve. Figure 4.24 is a good example of this problem. The smooth curve is totally different from one graph to the next. Results may be obtained fast, but a user wants good results.

In conclusion, even though the advanced smoothers are fast, they do not perform as well as the LOWESS smoother or the faster Cosine-Weighted Moving Average smoother in depicting simple functional relationships. The consistently good sum of squared residuals values and smooth curves produced by these simple smoothers favor their use over the advanced smoothers.

V. EVALUATION/APPLICATION OF THE ADVANCED SMOOTHING ALGORITHMS

A. GENERAL

The evaluation of Supersmoother and Split Linear Fit continues in this chapter. As stated in the previous chapter, a smoothing algorithm is used to extract the underlying relationship from a data set. Smoothing is especially useful if the underlying relationship is complex, i.e. too difficult to describe mathematically or use simple (global) least squares regression. The small and simple data sets in the previous chapter are easy to smooth since the shape of the underlying function is quite visible in a scatterplot of the raw data, see Figures 4.1, 4.7, and 4.16. Least squares regression is thus easy to apply to these data sets. On the other hand, the least squares method does not adequately smooth the data set tabulated in table form in Appendix D. A plot of this data is shown in Figure 1.1. The great amount of variability inherent in the data set causes the regression technique to be inadequate, i.e. the raw data is very erratic. Most data sets collected from real populations/situations do not have a constant nor smooth variance, thus regression techniques fail to be adequate, i.e. the fitted curve is too smooth and the sum of squared residuals is too high.

In this chapter the data set utilized is the daily sea-surface temperatures at Granite Canyon, just south of Point Sur, California [Ref. 10]. The data displayed in Figure 1.1 is the first of thirteen years of sea-surface temperature data collected at this location. This data set definitely does not have a constant variance, but it seems to exhibit some periodicity and to have some points of discontinuity, notably a very sudden and strong drop in temperature because of current up wellings in the spring. This data set was selected for final evaluation of Supersmoother and Split Linear Sit because of the following reasons:

1. this data set has been a subject of intense data analysis [Ref. 4];

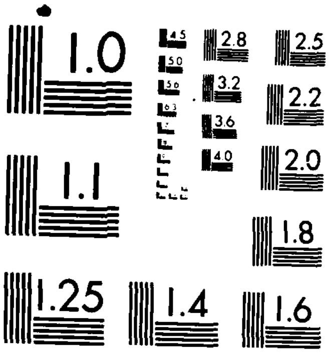
AD-A162 305 ANALYSIS OF TWO ADVANCED SMOOTHING ALGORITHMS(U) NAVAL 2/2
POSTGRADUATE CCHOOL MONTEREY CA J A VASQUEZ SEP 85

UNCLASSIFIED F/G 12/1 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

2. the characteristics exhibited by this data set, mentioned above, and;
3. it was easily accessible.

B. METHODOLOGY

The procedure followed in this chapter in the evaluation of Supersmoother and Split Linear Fit is described in the previous chapter in the Methodology section. The procedure is outlined below:

1. display and examine the data set to be smoothed;
2. display and examine the smooth results produced by the baseline smoothing techniques, i.e. the Least Squares Regression, the Equal-Weight Moving Average, the Cosine-Weighted Moving Average, and LOWESS;
3. display and examine the smooth results produced by the advanced smoothers;
4. compare these results to the results from 2 above.

In the previous chapter the graphs showing the smooth points produced by any one smoothing technique were displayed together, e.g. see Figure 4.3 which has all the Test Set One LOWESS smoothing results. In this chapter it is best to display together the smoothing results that use equivalent neighborhood sizes, as described below:

1. the neighborhood size in the Moving Average smoothing technique is called M, which is equivalent to window size used in Split Linear Fit;
2. the span value used by Supersmoother is equivalent to the F value used by LOWESS, both are computed as the ratio of the neighborhood size to the number of data points to be smoothed.

This change in plot display makes it easier to subjectively decide the adequacy or usefulness of the advanced smoothing algorithms. The word 'subjectively' is used as the measure of effectiveness because, as mentioned before, the decision to use one smooth curve over another is basically based on the user's needs and desires and on the curve's appearance. In order to bare the comparison a more concrete statistical analysis, the sum of squared residuals of the different plots will be calculated.

The complete sea-surface temperature data set contains 4380 data points, i.e. a sea-surface temperature corresponding to each day for the period of March 1, 1971 to February 1983, excluding the dates of February 29. Only the first 671 data points which correspond to 1971 and 1972 will be used in the evaluation in this chapter so as to have a manageable input data set.

Figure 5.1 shows a scatterplot of the data set used in this chapter. The axis scales shown in this figure will be the standard axis scales to be used in all the graphs corresponding to this chapter. The vertical axis which is easiest to describe displays the sea-surface temperature range in degrees centigrade. The horizontal axis displays the day that the temperature was measured. The numbers shown indicate the Calendar date, i.e. the first digit indicates the year, assuming that the corresponding decade is known. Recall that the abbreviated data set used in this chapter was collected in the 1970's. The next three digits indicate the day of the year, for example '080' means the 80th day of the year. The vertical dashed grid lines indicate the change in season during the year, e.g. at 1080 winter 1971 ends and spring 1971 begins. The solid tic marks indicate the end of a month. All the graphs begin with January. This is the reason that in Figure 5.1 the first two bins, January and February, are empty since the data set begins with March 1, 1971. Finally, since the abscissa are measured in days, the neighborhood sizes used in this chapter will also be in days, e.g. a neighborhood size of 5 will correspond to a period of 5 days.

C. TESTING AND RESULTS

Figure 5.1 exhibits the data set that is evaluated in this chapter. From this graph alone, it could be deduced that there are two temperature cycles present in the data set. The first cycle peaks near the end of summer 1971, and second cycle peaks just after the beginning of fall 1972. Another point of view that the analyst may take by looking at Figure 5.1 is to analyze the point distribution between the peaks,e.g. the data points from beginning of fall 1971 to beginning of fall 1972. Within this period the temperature appears to follow a cyclic process, but with a smaller period, e.g. from 1355 to 2080 the temperature increased and then decreased and from 2080 to 2172 the opposite is reflected. Therefore, one analyst may try to depict the intra-annual cycle, while another may try to display the inter-annual cycle, while still another may try to show both cycles or something else. The analysis in this chapter will concentrate on the intra-annual cycle as depicted by the advanced smoothers and compare these results to curves produced by previously validated and well-accepted smoothing techniques.

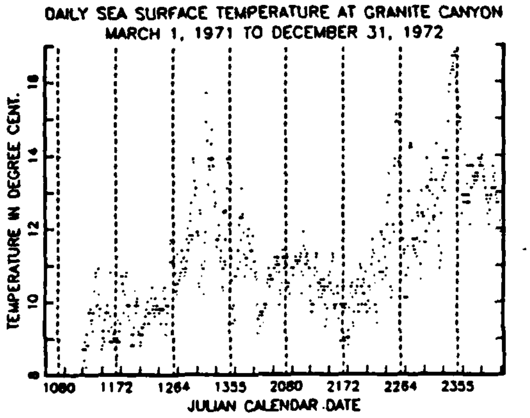


Figure 5.1 Data Set For Practical Application.

Figure 5.2 shows the data divided into the two annual periods in order to display the intra-annual characteristics of the data. With these displays the intra-annual variability is more detectable and a different view of smoothing the data can be taken, i.e. the data can be smoothed to show the cyclic effect within the seasons, for example between 1172 and 1264.

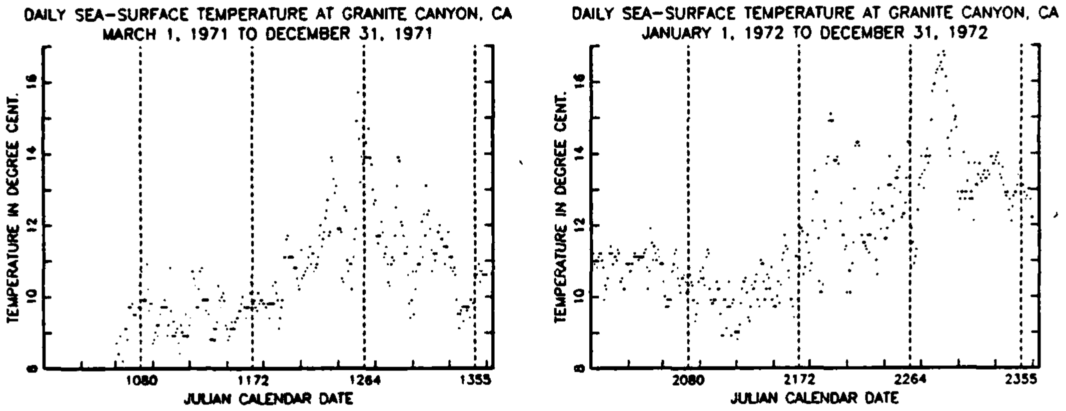


Figure 5.2 Data Set For Practical Application Divided Into Two Parts.

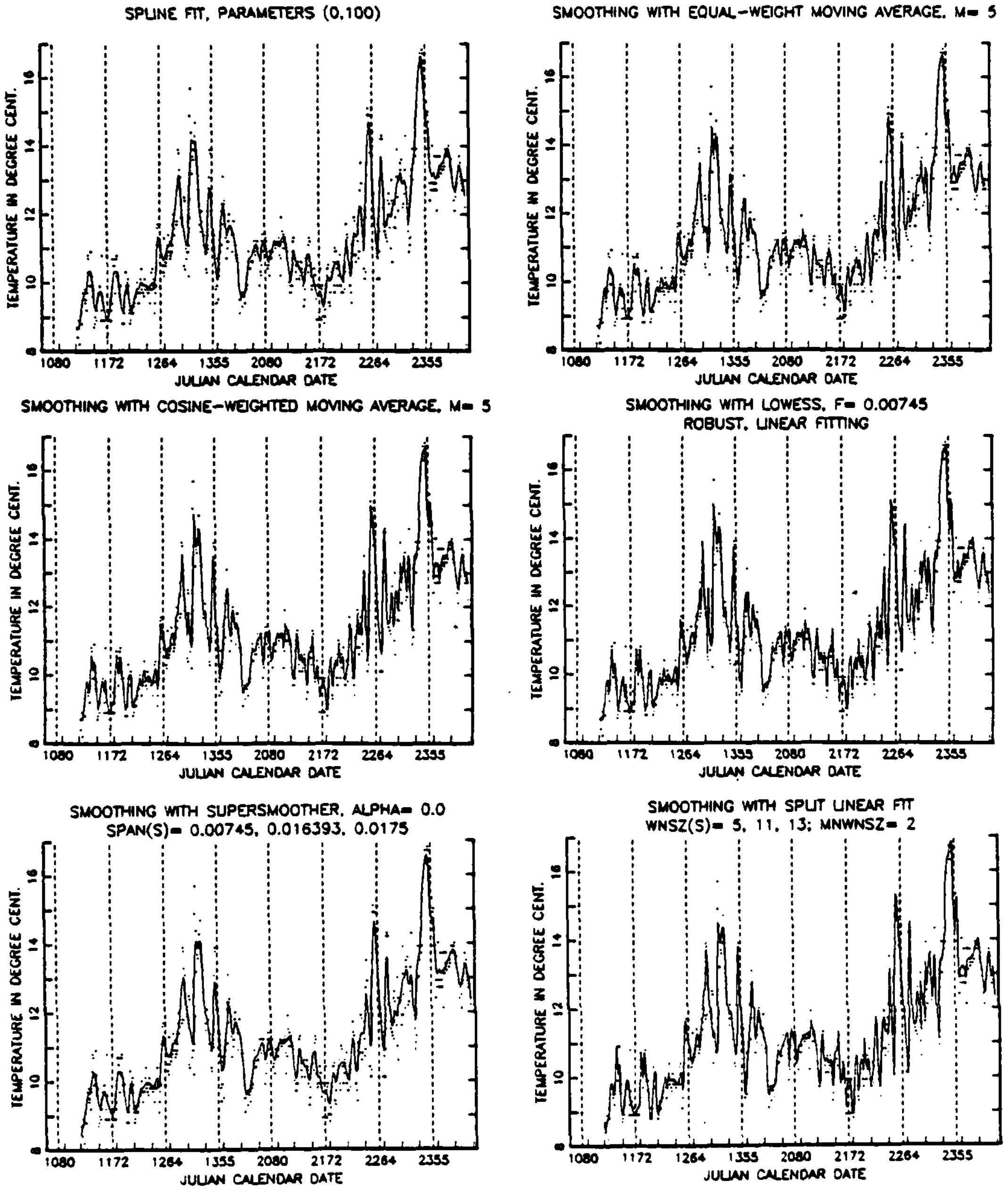
Figures 5.3 through 5.8 show the smooth curves produced by the smoothers being compared in this chapter. A different neighborhood size was used to smooth the data in each case. One of the three span values used in the Supersmoother corresponds to the neighborhood size that is used by the other smoothers within a figure; the same applies to the Split Linear Fit. In order to properly evaluate the adequacy of the smoothing produced by the advanced smoothers, it is best to compare similar smooth curves produced by both the advanced smoothers and the baseline smoothers.

As the neighborhood size increases, the produced smooth curve gets smoother. This effect is described by the analytical equation 1.4. In addition, as the neighborhood size increases, the Moving Average type smoothers loose more smooth data points from each end; this effect was described in Chapter I.

After each figure is a table which compares the sum of squared residuals corresponding to the curves displayed in the figure. This gives the analyst a better and more statistically supported comparison between the smooth output produced by the different smoothers. In addition, a table showing the CPU time used by each of the smoothers follows the sum of squared residuals table.

Figure 5.3 presents the smooth curves which are produced when the neighborhood size is small,i.e. neighborhood size of 5. The curves are very erratic and somewhat unpleasant to the eye. Because some of the peaks are too close to each other, the short term cyclic effect is overemphasized and rendered useless. In Table 7 it can be seen that the sum of squared residuals that correspond to the advanced smoothers are not as low as those corresponding to the simple smoothers. In fact the sum of squared residuals corresponding to the Supersmoother is almost three times that produced by LOWESS, which is the lowest value. Table 8 shows that the Cosine-Weighted Moving Average smoother is about three times faster than Supersmoother and about 5 times faster than Split Linear Fit, yet more accurate.

In Figure 5.4 the smooth plots are not as jagged as the ones shown in Figure 5.3. The reason for this difference is that the neighborhood size used to produce the smooth curves in Figure 5.4 is twice that of those used to produce the smooth curves in Figure 5.3. There are some slight differences between each smooth curve in Figure 5.4, but the differences are difficult to detect. The curve produced by Supersmoother seems to have the least amount of jagged peaks thus making it easier to count the increasing and decreasing cycles within each season. However, Table 9 shows that the sum of squared residuals produced by Supersmoother is not as good as those produced by the LOWESS, either robust or non-robust smooth curve. The analyst in this case has the choice of deciding whether to have a good smooth curve with a high sum of squared residuals or a not so smooth curve with a low sum of squared residuals. Table 10 shows that the Cosine-Weighted Moving Average smoother is much faster than either of



SPLINE FIT, PARAMETERS (0,100)

SMOOTHING WITH EQUAL-WEIGHT MOVING AVERAGE, M= 5

SMOOTHING WITH COSINE-WEIGHTED MOVING AVERAGE, M= 5

SMOOTHING WITH LOWESS, F= 0.00745
ROBUST, LINEAR FITTING

SMOOTHING WITH SUPERSMOOTHER, ALPHA= 0.0
SPAN(S)= 0.00745, 0.016393, 0.0175

SMOOTHING WITH SPLIT LINEAR FIT
WNSZ(S)= 5, 11, 13; MNWNSZ= 2

Figure 5.3 Comparison of Smoothers Using Neighborhood Size of 5.

TABLE 7

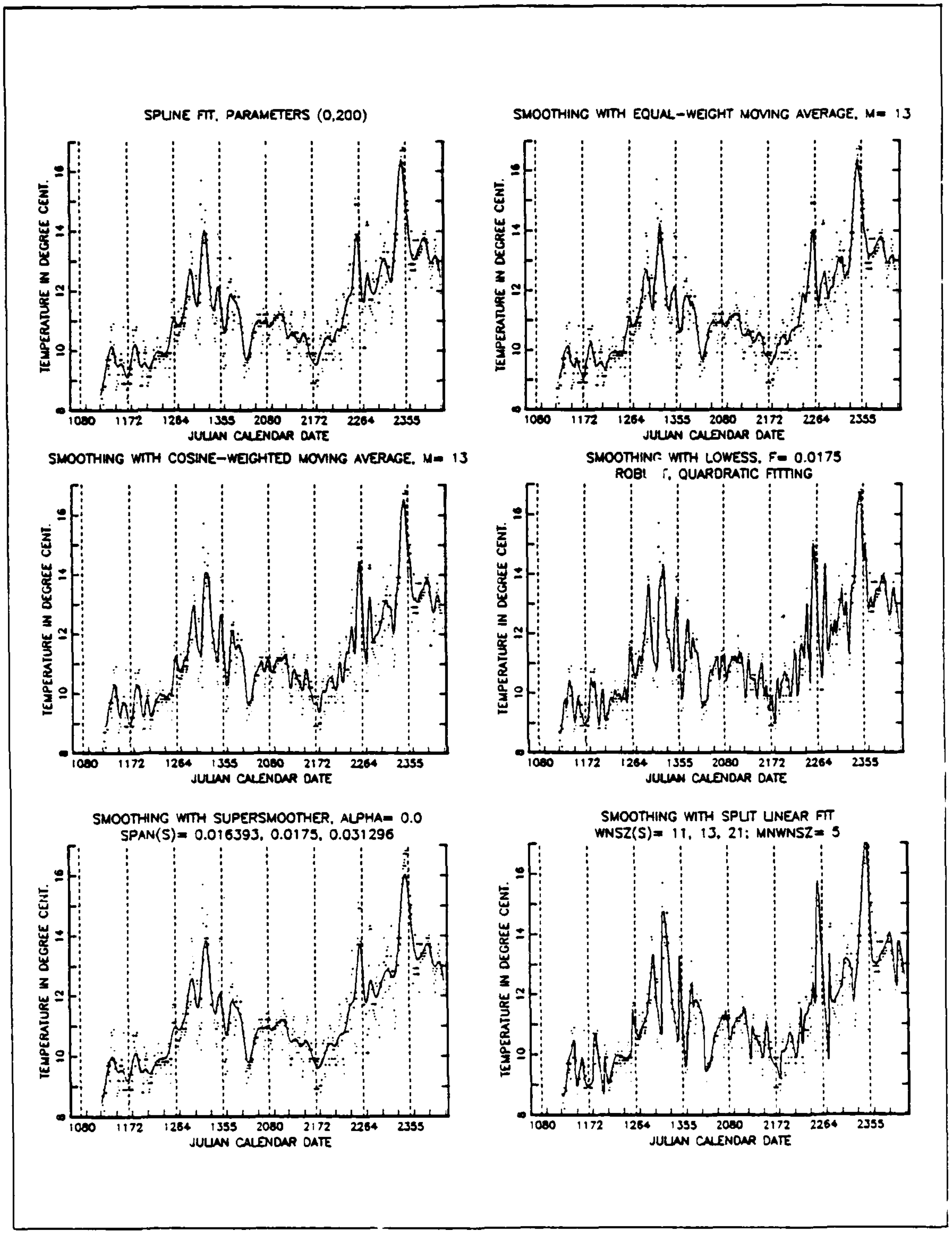
SUM OF SQUARED RESIDUALS USING NEIGHBORHOOD SIZE OF 5

Type of Fit	Sum of Squared Residuals
Spline (0,100)	100.47089
Equal-Weight Moving Average, M= 5	84.9068
Cosine-Weighted Moving Average, M= 5	48.0688
LOWESS, robust, F= 0.00745	49.1153
LOWESS, non-robust, F= 0.00745	37.3108
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.00745, 0.016393, 0.0175	105.7947
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 5, 11, 13	60.5805

TABLE 8

CPU USAGE: NEIGHBORHOOD SIZE OF 5

Type of Fit	CPU Consumed (in Seconds)
Spline (0,100)	26.6
Equal-Weight Moving Average, M= 5	0.03
Cosine-Weighted Moving Average, M= 5	0.69
LOWESS, robust, F= 0.00745	16.07
LOWESS, non-robust, F= 0.00745	5.41
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.00745, 0.016393, 0.0175	2.27
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 5, 11, 13	3.79


SPLINE FIT, PARAMETERS (0,200)

SMOOTHING WITH EQUAL-WEIGHT MOVING AVERAGE, M= 13

SMOOTHING WITH COSINE-WEIGHTED MOVING AVERAGE, M= 13

SMOOTHING WITH LOWESS, F= 0.0175
ROBUST, QUARDRATIC FITTING

SMOOTHING WITH SUPERSMOOTHER, ALPHA= 0.0
SPAN(S)= 0.016393, 0.0175, 0.031296

SMOOTHING WITH SPLIT LINEAR FIT
WNSZ(S)= 11, 13, 21; MNWNSZ= 5

Figure 5.4 Comparison of Smoothers Using Neighborhood Size of 13.

TABLE 9

SUM OF SQUARED RESIDUALS USING NEIGHBORHOOD SIZE OF 13

Type of Fit	Sum of Squared Residuals
Spline (0,200)	200.51872
Equal-Weight Moving Average, M= 13	236.86018
Cosine-Weighted Moving Average, M= 13	134.92327
LOWESS, robust, F= 0.0175	86.4623
LOWESS, non-robust, F= 0.0175	74.3993
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.016393, 0.0175, 0.031296	233.94982
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 11, 13, 21	186.25445

TABLE 10

CPU USAGE: NEIGHBORHOOD SIZE OF 13

Type of Fit	CPU Consumed (in Seconds)
Spline (0,200)	38.0
Equal-Weight Moving Average, M= 13	0.04
Cosine-Weighted Moving Average, M= 13	0.73
LOWESS, robust, F= 0.0175	12.02
LOWESS, non-robust, F= 0.0175	3.95
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.016393, 0.0175, 0.031296	2.32
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 11, 13, 21	3.77

the advanced smoothers in addition to being the most accurate smoother.

With a neighborhood size of little less than a month, i.e. 21 which is equivalent to 21 days since time is the unit of measurement of the abscissa in this data set, most of the smoothers produce smooth curves which have lost the jagged effect at the high and low points of the smooth curves, see Figure 5.5. In this figure, the smooth curve produced by Split Linear Fit displays its tendency to follow and emphasize outliers. Split Linear Fit results are so robust that the peaks are shown pointed and not round as displayed by the other smoothers. This effect is caused by the edge-detection weighting scheme of Split Linear Fit. As shown in Table 11, the sum of squared residuals produced by the advanced smoothers are higher than most of the other smoothers, even though a very similar smooth curve is produced by all the smoothers. The difference between the sum of squared residuals value corresponding to Split Linear Fit and Supersmoother, respectively, can be explained by the tendency of Split Linear Fit to follow outliers more closely than Supersmoother. The Split Linear Fit smoother lets the raw data dictate the shape of the smooth curve, therefore, the difference between the raw data and the smoothed data is smaller for the Split Linear Fit than for the Supersmoother. Table 12 shows again that the Cosine-Weighted Moving Average smoother is faster, even though the sum of squared residuals value may not be the best. The Supersmoother and the Split Linear Fit smoothers have consistently maintained their usage of CPU.

Figure 5.6 displays the results produced by using a neighborhood size equivalent to almost one month, i.e. 29 days. Though still quite similar, each smoother produces a visibly different smooth curve. The only exception is Equal-Weight Moving Average smoother which has suppressed the influence of the outliers. The shape of the input data is still being maintained by most of the smoothers, especially Split Linear Fit which is designed to do so. In Table 13, the sum of squared residuals values corresponding to the advanced smoothers do

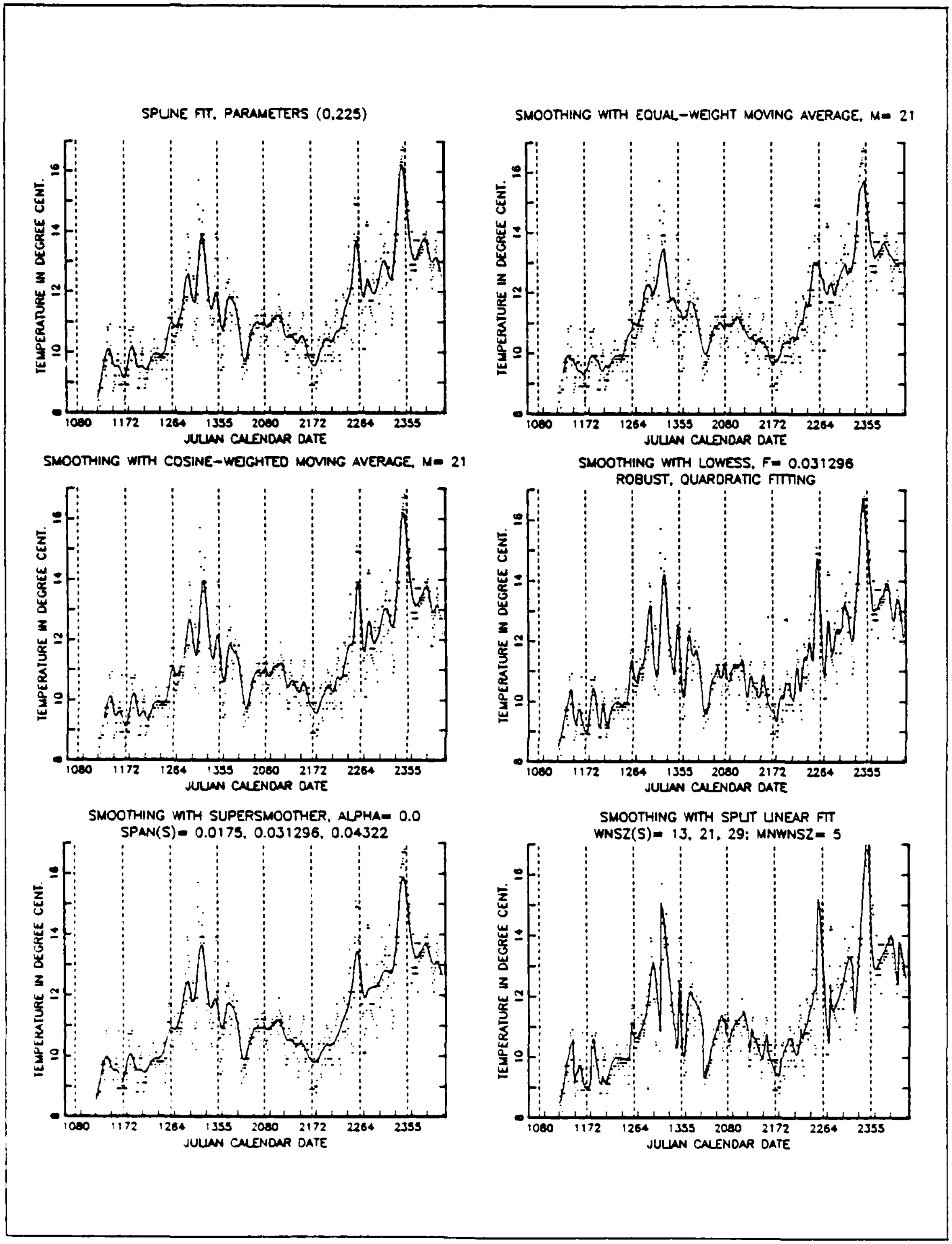


Figure 5.5 Comparison of Smoothers Using Neighborhood Size of 21.

TABLE 11

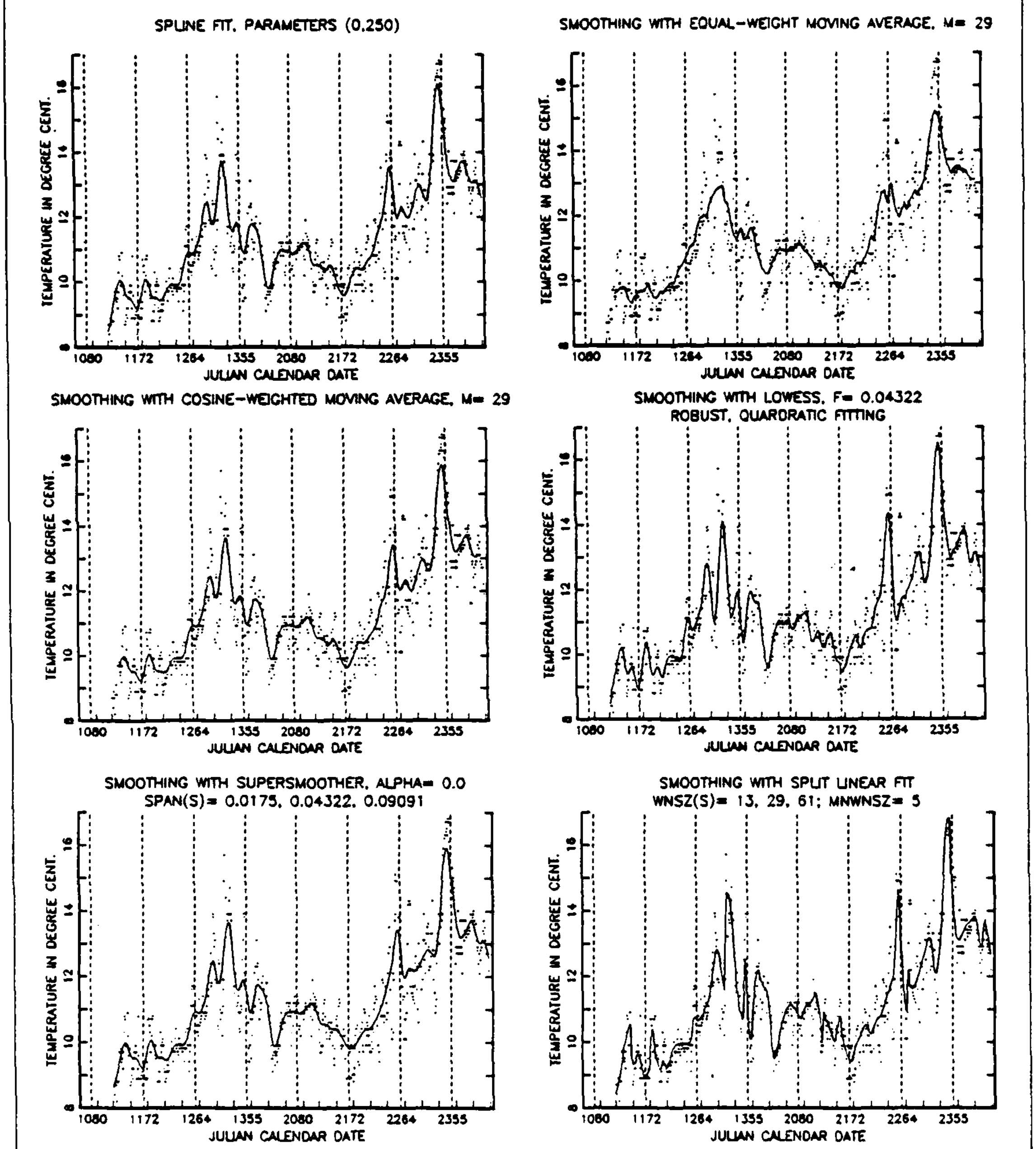
SUM OF SQUARED RESIDUALS USING NEIGHBORHOOD SIZE OF 21

Type of Fit	Sum of Squared Residuals
Spline (0,225)	225.04812
Equal-Weight Moving Average, M= 21	331.967483
Cosine-Weighted Moving Average, M= 21	209.75158
LOWESS, robust, F= 0.031296	162.382
LOWESS, non-robust, F= 0.031296	142.891
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.0175, 0.031296, 0.04322	270.39113
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 13, 21, 29	236.34410

TABLE 12

CPU USAGE: NEIGHBORHOOD SIZE OF 21

Type of Fit	CPU Consumed (in Seconds)
Spline (0,225)	43.87
Equal-Weight Moving Average, M= 21	0.07
Cosine-Weighted Moving Average, M= 21	0.76
LOWESS, robust, F= 0.031296	29.56
LOWESS, non-robust, F= 0.031296	10.03
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.0175, 0.031296, 0.04322	2.25
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 13, 21, 29	3.7



SPLINE FIT, PARAMETERS (0.250)

SMOOTHING WITH EQUAL-WEIGHT MOVING AVERAGE, M= 29

SMOOTHING WITH COSINE-WEIGHTED MOVING AVERAGE, M= 29

SMOOTHING WITH LOWESS, F= 0.04322
ROBUST, QUARDRATIC FITTING

SMOOTHING WITH SUPERSMOOTHER, ALPHA= 0.0
SPAN(S)= 0.0175, 0.04322, 0.09091

SMOOTHING WITH SPLIT LINEAR FIT
WNSZ(S)= 13, 29, 61; MNWNSZ= 5

Figure 5.6 Comparison of Smoothers Using Neighborhood Size of 29.

TABLE 13

SUM OF SQUARED RESIDUALS USING NEIGHBORHOOD SIZE OF 29

Type of Fit	Sum of Squared Residuals
Spline (0,250)	249.84103
Equal-Weight Moving Average, M= 29	411.15847
Cosine-Weighted Moving Average, M= 29	266.83401
LOWESS, robust, F= 0.04322	235.881
LOWESS, non-robust, F= 0.04322	202.825
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.0175, 0.04322, 0.09091	265.40284
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 13, 29, 61	211.79171

TABLE 14

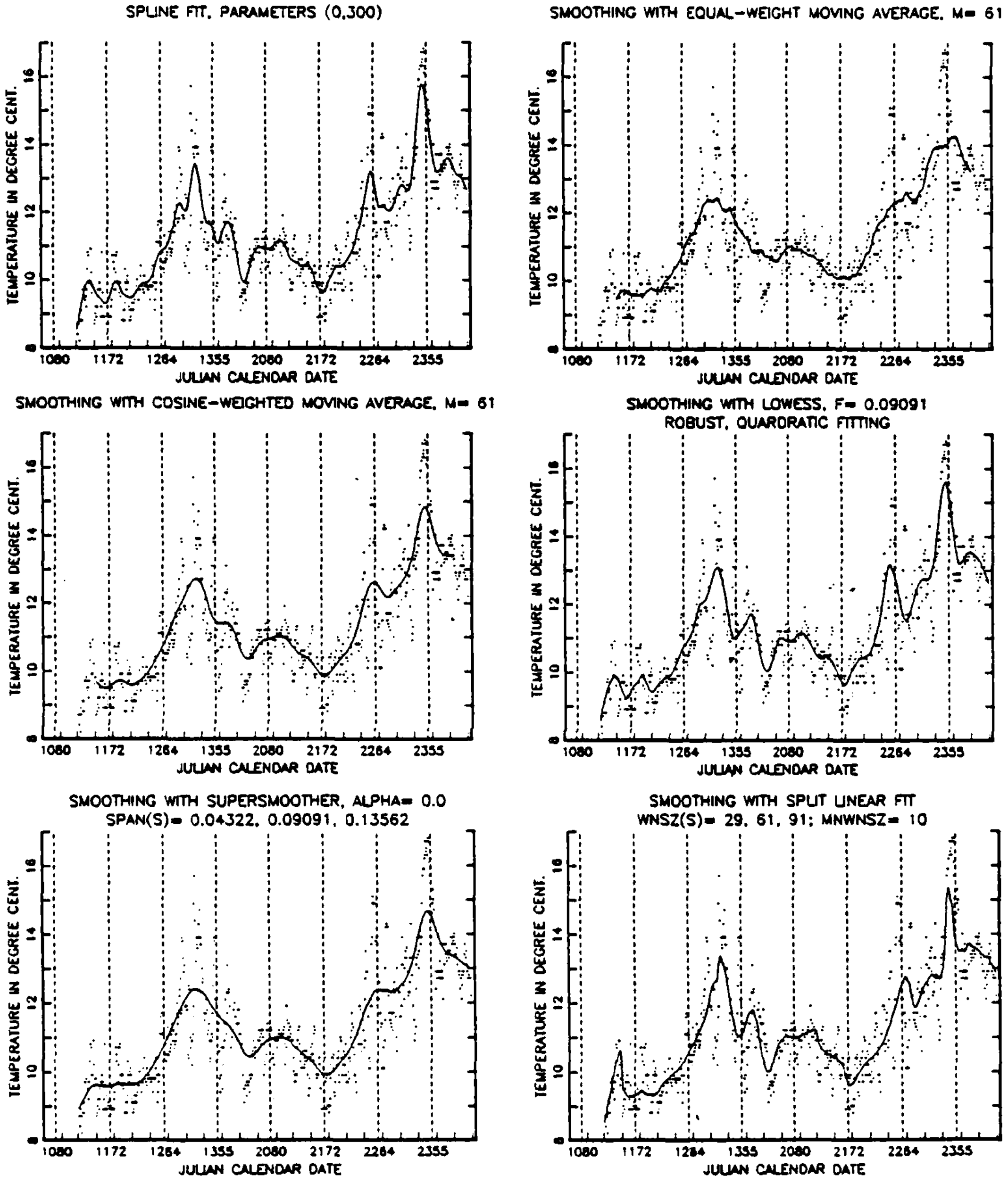
CPU USAGE: NEIGHBORHOOD SIZE OF 29

Type of Fit	CPU Consumed (in Seconds)
Spline (0,250)	42.55
Equal-Weight Moving Average, M= 29	0.08
Cosine-Weighted Moving Average, M= 29	0.81
LOWESS, robust, F= 0.04322	34.21
LOWESS, non-robust, F= 0.04322	11.28
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.0175, 0.04322, 0.09091	2.28
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 13, 29, 61	3.78

not deviate as much as in the three past cases from the sum of squared residuals values corresponding to the other smoothers. Table 14 basically follows the same explanation of Table 12.

If a neighborhood of two months is used, the smooth curves shown in Figure 5.7 are produced. Each curve is now quite different from the other curves. It is now quite noticeable that the Moving Average type smoothers have lost smooth data points at the ends. LOWESS has been able to maintain the shape of the input data. The Split Linear Fit has made a good attempt to do as well with the edge-detecting weighting scheme. Since Supersmoother is a central smoother and the neighborhood size is larger than the intra-seasonal period, the smooth curve produced is quite 'smooth', i.e. not jagged and abrupt. The sum of squared residuals values shown in Table 15 reflect the superiority of LOWESS over the other smoothers. The smooth curves displayed in Figure 5.7 substantiate even more LOWESS's superior performance. LOWESS has a better sum of squared residuals value and a better smooth curve. Table 16 shows that LOWESS is almost the slowest of the smoothing techniques, but it has the lowest sum of squared residuals.

The last figure, Figure 5.8, is shown basically to illustrate that the Moving Average smoothers have begun to deteriorate, i.e. loose too many smooth data points at the ends and deviate from the shape of the raw data. LOWESS and the advanced smoothers are still maintaining the general shape of the raw data, but are beginning to get too smooth. The Split Linear Fit smoother has made a good attempt to depict the outliers, even with a large neighborhood size, but the price paid is the return of the undesirable sharp peaks with plateau-like bases. If the neighborhood size is gradually increased, the resulting smooth curves will change from those in Figure 5.8 to smooth sinusoidal curves, and eventually to straight lines. The sinusoidal curves and the straight lines illustrate only general features about the raw data and defeat the purpose of smoothing. As shown in Table 17, where the neighborhood size is 91 days, the advanced smoothers finally produced the low sum of squared residuals values. This figure was made to illustrate that the


SPLINE FIT, PARAMETERS (0,300)

SMOOTHING WITH EQUAL-WEIGHT MOVING AVERAGE, M= 61

SMOOTHING WITH COSINE-WEIGHTED MOVING AVERAGE, M= 61

SMOOTHING WITH LOWESS, F= 0.09091
ROBUST, QUARDRATIC FITTING

SMOOTHING WITH SUPERSMOOTHER, ALPHA= 0.0
SPAN(S)= 0.04322, 0.09091, 0.13562

SMOOTHING WITH SPLIT LINEAR FIT
WNSZ(S)= 29, 61, 91; MNWNSZ= 10

Figure 5.7 Comparison of Smoothers
Using Neighborhood Size of 61.

TABLE 15

SUM OF SQUARED RESIDUALS USING NEIGHBORHOOD SIZE OF 61

Type of Fit	Sum of Squared Residuals
Spline (0,300)	299.19197
Equal-Weight Moving Average, M= 61	517.92528
Cosine-Weighted Moving Average, M= 61	409.40077
LOWESS, robust, F= 0.09091	372.995
LOWESS, non-robust, F= 0.09091	372.263
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.04322, 0.09091, 0.13562	453.02390
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 29, 61, 91	423.35262

TABLE 16

CPU USAGE: NEIGHBORHOOD SIZE OF 61

Type of Fit	CPU Consumed (in Seconds)
Spline (0,300)	46.17
Equal-Weight Moving Average, M= 61	0.14
Cosine-Weighted Moving Average, M= 61	0.98
LOWESS, robust, F= 0.09091	51.78
LOWESS, non-robust, F= 0.09091	17.44
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.04322, 0.09091, 0.13562	2.28
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 29, 61, 91	3.8

advanced smoothers maintain the shape of the raw data better than the other smoothers when larger neighborhood sizes are used in the smoothing. Table 18 illustrates that the advanced smoothers maintain a constant CPU usage, throughout the evaluation. Tables 8 through 18 have larger CPU usage than in the previous chapter. The reason for this is that the data sizes are different.

D. CONCLUSIONS

The two advanced smoothers investigated in this thesis, the Supersmoother and the Split Linear Fit, generate adequate smooth curves. They are faster than most current smoothing techniques. However, their many inputs make their implementation difficult.

Two simpler smoothers are the LOWESS and the Cosine-Weighted Moving Average. Both only require a single neighborhood size as input. This dramatically reduces the complexity of the program for the user. Both generate smooth curves with satisfactory results equal to the advanced smoothers. However, both LOWESS and the Cosine-Weighted Moving Average produce better sum of squared residuals values. In addition, the Cosine-Weighted Moving Average is much faster than either of the advanced smoothers.

The simpler smoothers, LOWESS and the Cosine-Weighted Moving Average, do have some drawbacks. LOWESS is considerably slow than the advanced smoothers, but the disadvantage of LOWESS is only apparent after many runs of the programs. The speed difference for a single run is minor, measured only in seconds. The disadvantage of the Cosine-Weighted Moving Average smoother is that values are dropped from the ends of the output array, as illustrated in this thesis. The larger the neighborhood size, the more smoothed values are dropped, sometimes these values are important and other times they are not; this decision belongs to the user.

It is the recommendation of the author that LOWESS or the Cosin-Weighted Moving Average smoother be used over either of the advanced smoothers. The advanced smoothers are considerably more

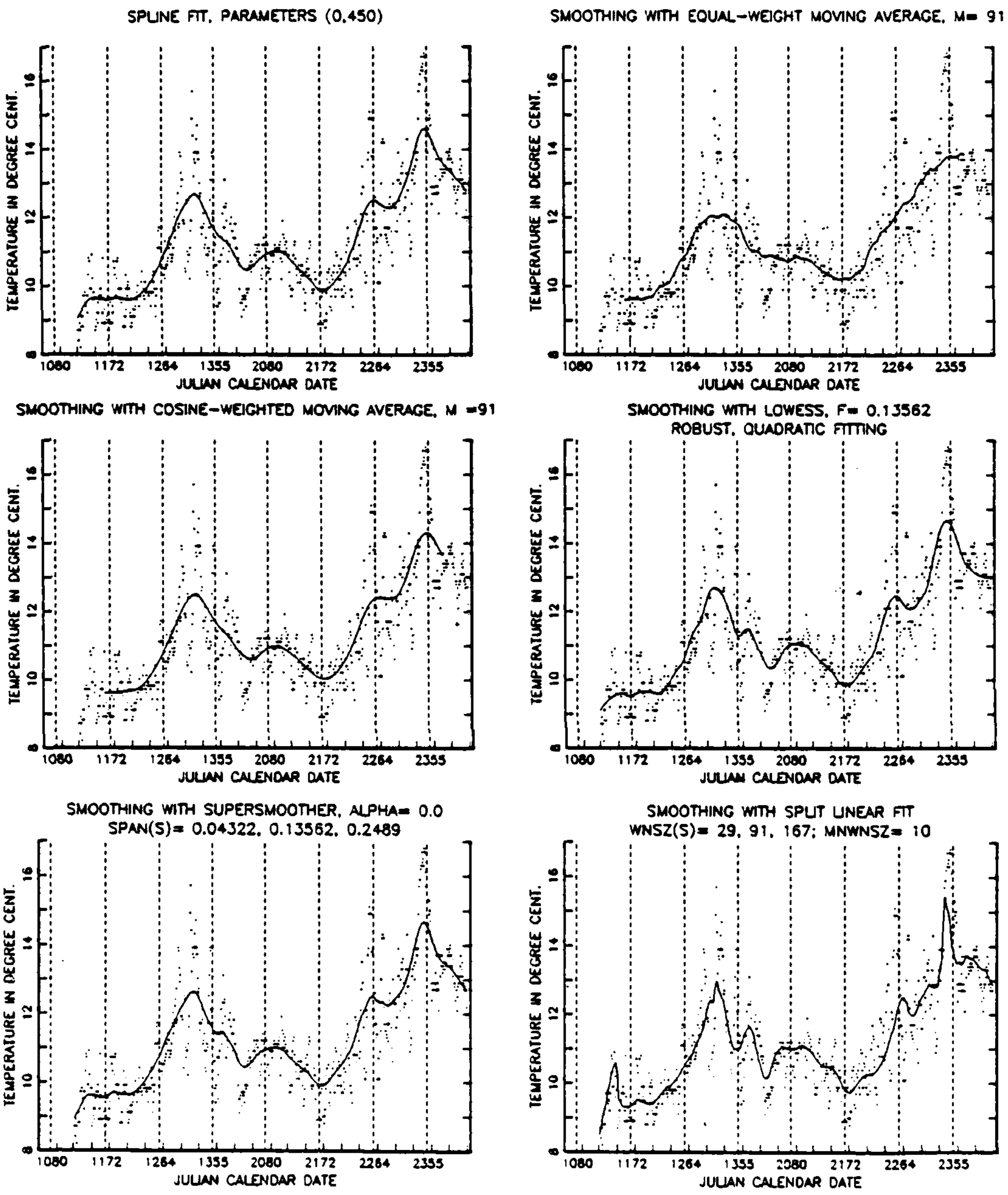


Figure 5.8 Comparison of Smoothers Using Neighborhood Size of 91.

TABLE 17

SUM OF SQUARED RESIDUALS USING NEIGHBORHOOD SIZE OF 91

Type of Fit	Sum of Squared Residuals
Spline (0,450)	450.54580
Equal-Weight Moving Average, M= 91	550.83383
Cosine-Weighted Moving Average, M= 91	463.34429
LOWESS, robust, F= 0.13562	458.317
LOWESS, non-robust, F= 0.13562	454.213
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.04322, 0.13562, 0.2489	435.71549
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 29, 91, 167	452.55936

TABLE 18

CPU: USAGE NEIGHBORHOOD SIZE OF 91

Type of Fit	CPU Consumed (in Seconds)
Spline (0,450)	55.27
Equal-Weight Moving Average, M= 91	0.2
Cosine-Weighted Moving Average, M= 91	1.1
LOWESS, robust, F= 0.13562	68.04
LOWESS, non-robust, F= 0.13562	23.01
Supersmoother, ALPHA= 0.0 SPAN(s)= 0.04322, 0.13562, 0.2489	2.27
Split Linear Fit, MNWNSZ= 2 WNSZ(s)= 29, 91, 167	3.82

complex than these smoothers without yielding better accuracy. As mentioned before, the speed difference is minor, measured only in seconds.

VI. INSTRUCTIONS ON USING THE ADVANCED SMOOTHERS

A. GENERAL

This chapter provides detailed instructions on how to use the smoothing programs developed by the author of this thesis and have the advanced smoothing algorithms embedded in them. The Supersmoother algorithm is embedded in one smoothing program and the Split Linear Fit algorithm is embedded in another smoothing program. Any interested person should be familiar with this chapter before attempting to do any data smoothing with these programs. In order to obtain good and fast results and understand the smoothing algorithms, it is highly recommended that the user read either or both of Chapters II and III, depending on the program to be used. Before adjusting any embedded parameters, it is essential that the user read the 'Technical Description' chapter corresponding to the program being modified. These programs are designed to be used on the IBM 3033 computer currently at the Naval Postgraduate School. The programs are written in FORTRAN 77, because of the need to use negative index values. The use of both of the smoothing programs is very similar, so both are addressed in this chapter. Operations peculiar to each program are addressed as separate paragraphs corresponding to each program.

The smoothing programs are completely interactive, in other words, the user enters the data and other pre-defined parameters when asked by built-in queries. The user has the option of selecting the one of several types of output, which are:

1. create a CMS file and place the smooth output into this newly created CMS file;
2. place the smooth output into an existing APL workspace within a newly created APL variable;
3. create an APL workspace and place the smooth output into this newly created workspace, or;
4. plot the smooth output using the GRFASTAT graphics package.

The smoothing programs are quite flexible in letting the user decide where to put the final smooth output. The user can smooth any number of points up to 5000 data points. The data to be smoothed need not be in order, because the smoothing programs have an embedded sorting subroutine which sorts the data into chronological according to the abscissa values entered by the user, before sending the data array to the advanced smoothing algorithms.

The programs are written in such a way that if the user makes a mistake, it will be announced, and the program can be restarted or stopped. Practically no knowledge of FORTRAN is needed to run these programs. If APL is to be used, it is best to understand what the relevant 'workspace' and 'variable' names are [Ref. 15]. If the user wants to use GRAFSTAT to plot the output, it is best to get familiar with the GRAFSTAT 'PLOT' and 'AXIS CONTROL' functions before attempting to use the plotting option embeded within the advanced smoothing programs.

B. TERMINAL REQUIREMENTS

If used to create a CMS data file the Supersmoothing program is used to can be run from any remote terminal attached to the IBM 3033 and located within the Naval Postgraduate School. If this smoothing program is to be used for APL workspace creation, then an appropriate APL terminal must be used. If the GRAFSTAT plotting option is to be used, then the IBM 3277/TEK 618 graphics terminals must be used.

Because the Split Linear Fit generates a great amount of data it must be run on the IBM 3277/TEK 618 graphics terminals with a memory capacity of at least 2 Mega-Bytes. The bigger the input data set, the more data storage that the Split Linear Fit smoothing program will need. For each point that is to be smoothed, the computer needs the capacity to store a matrix that has the dimensions of 9 by the number window sizes entered. For example, if 200 data points are to be smoothed with the Split Linear Fit smoothing program and 6 window sizes are entered, then each data point will need a matrix of size 9 by 6. Therefore, to run this smoothing program, the user must have available the storage capacity for a matrix that is 200 by 9 by 6, i.e. 10,800 bytes, plus the storage capacity for the raw data, the corresponding abscissa and computed weights, the temporary smoothed point values, and the final smoothed point values.

C. INPUT DATA FILES

In order to use either of the smoothing programs, it is required that the data which is to be smoothed be in a CMS file with filetype 'data'. If the data points are not in chronological order, the corresponding abscissa, i.e. numerical order, must be in another CMS file with filetype 'ORD'.

D. PROGRAM INITIALIZATION

The advanced smoothing program packages can be obtained from Professor P. A. W. Lewis, Department of Operations Research, U. S. Naval Postgraduate School, Monterey, CA. The Supersmoothing program consists of the following files:

1. SUPSMO EXEC A1;
2. SUPSMO FORTRAN A1;
3. SUPSMO VSAPLWS A1.

A copy of these files is in Appendix A. The Split Linear Fit program consists of the following files:

1. SPTLIN EXEC A1;
2. SPTLIN FORTRAN A1;
3. SPTLIN VSAPLWS A1.

A copy of each of these files is in Appendix B. It is essential that the three respective files be on the same disk when either smoothing program is to used. The EXEC file do the following operations:

1. activates the IBM;
2. system libraries;
3. queries the user for the input;
4. designates the computer storage space to be used for input and output;

5. loads and runs the FORTRAN file;
6. executes the APL or GRAFSTAT user options;
7. returns the disk to the original state, i.e. erases the TEXT and LOAD files, so as not to overload the disk being used.

The Supersmoother smoothing program is invoked by typing 'SUPSMO' and then pressing the ENTER key on the keyboard. Next the user must read the information displayed and comply with the instructions. As long as the user follows the instructions, the smoothing program 'SUPSMO' will produce the desired results. If any deviations from the requested data occur 'SUPSMO' will let the user know. The Split Linear Fit smoothing program is just as easy. It is invoked by typing 'SPTLIN' and pressing the ENTER key on the keyboard. Read the information on the screen, answer the questions, and 'SPTLIN' does the rest.

An example of a session using SUPSMO to create the Supersmoother curve in Figure 5.3 is in Appendix E. A session using the Split Linear Fit smoothing program SPTLIN basically follows the same line of questions.

E. OUTPUT FILES

The smoothing programs will put the smooth output where the user designates unless a file already exits with that name. If a CMS file already exists by the name that the user wants, then the session will be terminated, told the reason for the termination and to restart the program. If the CMS file does not exist then the program continues normally. For APL files, the program queries the user about the status of the file, i.e. exiting or to be created. If the new file exists or if old file does not exist, then the session is terminated, the user is told the reason for the termination and to restart. One word of caution: THE SMOOTHING PROGRAMS WILL NOT WRITE OVER AN EXISTING APL VARIABLE!!! The program will continue running normally, but the data will be lost. Therefore, it is up to the user to manage the disk space properly and to keep note of which file contains what type of smooth data. The CMS file created by the smoothing programs will contain five smooth data points per row. The length will depend on the number of points that are smoothed, i.e. the number of points smoothed divided by five.

In order to put data into an existing APL workspace or create a new workspace, the user types in the name of the APL workspace when asked to do so. The will verify the status of the workspace as mention before. If everything is satisfactory, the program continues.

Should the user have any specific questions about the programs it is recommended that the 'TECHNICAL DESCRIPTION' chapter be read. These chapters basically follow the smoothing procedure step by step.

APPENDIX A

SUPERSMOOTHER PROGRAM

1. *SUPSMO EXEC*

The following computer file is SUPSMO EXEC which activates and runs the Supersmoother smoothing program. Chapter VI contains instructions on how to use this smoothing program.

```
&TRACE
SET BLIP *
GLOBAL  TXTLIB  VLNKMLIB  VALTLIB  VFORTLIB  IMSLSP  NONIMSL
CMSLIB
CLRSCRN
&TYPE YOU HAVE INITIATED AN ALGORITHM
&TYPE TO SMOOTH A SET OF DATA USING THE
&TYPE ALGORITHM "SUPER SMOOTHER"
&TYPE DEVELOPED BY FRIEDMAN AND STUETZLE OF
&TYPE STANFORD UNIVERSITY DEPT. OF STATISTICS
&TYPE
&TYPE IF GRAPHICS WILL NOT BE USED DEFINE STORAGE AS 1024K
&TYPE BY ENTERING 'DEF STOR 1024K'
&TYPE FOLLOWED BY 'I CMS',
&TYPE THEN BY 'SUPSMO'
&TYPE
&TYPE DO YOU WISH TO CONTINUE?
&TYPE ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:
&READ VAR &CONT
CLRSCRN
&IF &CONT NE Y &GOTO -EXIT
&TYPE IN ORDER TO USE THIS ALGORITHM
&TYPE YOU MUST HAVE ON HAND THE FOLLOWING:
&TYPE
&TYPE 1.   FILENAME OF DATA FILE (FILETYPE DATA) WITH
&TYPE      DATA TO BE SMOOTHED
&TYPE
&TYPE 2.   IF DATA POINTS ARE NOT IN CHRONOLOGICAL ORDER,
&TYPE      YOU NEED TO HAVE A FILE (FILETYPE ORDER)
&TYPE      WITH INDICES CORRESPONDING TO DATA POINTS
&TYPE      INDICATING THE ORDER OF THE DATA POINTS.
&TYPE
&TYPE 3.   FILENAME OF DATA FILE WHERE SMOOTHED OUTPUT
&TYPE      WILL BE WRITTEN OR IF YOU WANT
&TYPE      TO WRITE OUTPUT INTO APL HAVE ON HAND VARIABLE
&TYPE      AND WORKSPACE NAMES THAT WILL STORE THE OUTPUT.
&TYPE
&TYPE 4.   IF YOU WANT TO SMOOTH THE DATA USING ONLY
&TYPE      ONE WINDOW SIZE, HAVE ON HAND
&TYPE      THE DECIMAL FRACTION OF THE DATA TO BE USED.
&TYPE
&TYPE 5.   IF YOU WANT TO SMOOTH THE DATA USING
&TYPE      THREE WINDOW SIZES, HAVE ON HAND
&TYPE       THE  THREE  DECIMAL  FRACTIONS  OF  THE  DATA  TO  BE
USED.
&TYPE
&TYPE DO YOU WISH TO CONTINUE?
&TYPE ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:
&READ VAR &CONT
CLRSCRN
&IF &CONT NE Y &GOTO -EXIT
```

```
&TYPE ENTER FILENAME OF FILE WHICH CONTAINS
&TYPE THE DATA TO BE SMOOTHED:
&READ ARGS
&IF &N ≡ 0 &GOTO -TELL
&IF &N > 1 &GOTO -TELL
STATE &1 DATA A1
&IF &RC NE 0 &GOTO -ERROR
CLRSCRN
-ORDR CLRSCRN
&TYPE ARE DATA POINTS TO BE SMOOTHED
&TYPE IN CHRONOLOGICAL ORDER?
&TYPE ENTER Y FOR YES OR N FOR NO:
&READ VAR &CONT
&IF &CONT EQ Y &GOTO -GO
&TYPE SINCE THE DATA POINTS ARE NOT
&TYPE IN CHRONOLOGICAL ORDER, THEREFORE
&TYPE ENTER FILENAME OF FILE (FILETYPE ORDER)
&TYPE THAT CONTAINS ORDER INDICES
&READ VAR &ORD
STATE &ORD ORDER A1
&IF &RC NE 0 &GOTO -ERROR
-GO CLRSCRN
&TYPE THE DATA YOU WANT TO SMOOTH IS IN &1 DATA
&TYPE WHERE DO YOU WANT TO WRITE THE SMOOTHED OUTPUT?
&TYPE CMS OR APL?
-STRT &TYPE YOU CAN PLOT THE SMOOTHED OUTPUT
&TYPE IF YOU ARE LOGGED ON A TERMINAL
&TYPE THAT CAN ACCESS GRAFSTAT, I.E. HAVE 2M OF STORAGE
&TYPE BUT THE OUTPUT MUST BE STORED IN AN APL VARIABLE
&TYPE ENTER APL OR CMS:
&READ VAR &PLA
&IF &PLA EQ APL &GOTO -AP1
&TYPE
CLRSCRN
&TYPE THE SMOOTHED OUTPUT WILL BE WRITTEN
&TYPE TO A CMS FILE (FILETYPE DATA)
&TYPE ENTER ONLY THE FILENAME YOU WANT
&TYPE TO USE FOR THAT CMS FILE:
&READ VAR &FN
&TYPE THE SMOOTHED OUTPUT WILL BE WRITTEN
&TYPE INTO THE CMS FILE &FN DATA
&GOTO -COM
-AP1 &TYPE
&TYPE NOT USING THE NAME OF THE FILE
&TYPE WITH THE INPUT DATA, &1
&TYPE ENTER THE NAME OF THE APL VARIABLE
&TYPE THAT WILL STORE THE OUTPUT:
&READ VAR &A
&TYPE DO YOU WANT TO PLOT THE OUTPUT?
&TYPE ENTER Y FOR YES OR N FOR NO:
&READ VAR &GRF
&IF &GRF EQ Y &GOTO -PLOT
&TYPE ENTER THE NAME OF THE APL WORKSPACE
&TYPE THAT WILL CONTAIN &A :
&READ VAR &WKS
&TYPE IS &WKS AN EXISTING WORKSPACE OR A NEW WORKSPACE?
&TYPE ENTER O FOR EXISTING OR N FOR NEW:
&READ VAR &AGE
&FN ≡ TE
&GOTO -COM
-PLOT &TYPE CAN YOU ACCESS 2M OF STORAGE
&TYPE ON THIS DISK (TERMINAL)?
&TYPE ENTER Y FOR YES OR N FOR NO:
&READ VAR &GRF
&IF &GRF EQ N &GOTO -STRT
&FN ≡ TE
-COM CLRSCRN
&TYPE PLEASE READ THE FOLLOWING INSTRUCTIONS VERY CAREFULLY
&TYPE ARE YOU READY TO START THE SUPER SMOOTHING PROGRAM?
&TYPE ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:
```

```
&READ VAR &O
CLRSCRN
&IF &O NE Y &GOTO -EXIT
&TYPE PLEASE WAIT THE SMOOTHING PROGRAM IS BEING COMPILED
FI 04 CLEAR
FI 05 CLEAR
FI 06 CLEAR
FI 07 CLEAR
FI 08 CLEAR
FI 09 CLEAR
FORTVS SUPSMO (LVL (77))
FIL 04 DISK &ORD ORDER
FIL 07 DISK &1 DATA
FIL 08 DISK &FN DATA (RECFM FBA LRECL 80 BLKSIZE 800)
CLRSCRN
&TYPE PLEASE WAIT SMOOTHING PROGRAM IS BEING LOADED
LOAD SUPSMO (START
CLRSCRN
ERASE SUPSMO LISTING
ERASE SUPSMO TEXT
ERASE LOAD MAP
&IF &PLA EQ CMS &GOTO -EX
&IF &GRF EQ N &GOTO -NGRF
CP TERMINAL APL ON
&STACK )LOAD SUPSMO
&STACK &A ←CMSREAD
&STACK &FN
&STACK DATA
&STACK N
&STACK &A ← ,&A
&STACK &1 ←CMSREAD
&STACK &1
&STACK DATA
&STACK N
&STACK &1 ← ,&1
&STACK )SAVE
&TYPE ****PLEASE WAIT, LINKING TO GRAFSTAT*****************
&STACK )LOAD GRAFSTAT
&STACK DUM ←CMS 'CLRSCRN'
&STACK )PCOPY SUPSMO
&STACK ST RT
EXEC APLGS
&GOTO -DRP
-NGRF CP TERMINAL APL ON
&STACK )LOAD SUPSMO
&STACK &A ←CMSREAD
&STACK &FN
&STACK DATA
&STACK N
&STACK &A ← ,&A
&STACK )SAVE
&STACK )CLEAR
&IF &AGE EQ O &STACK )LOAD &WKS
&IF &AGE EQ N &STACK )WSID &WKS
&STACK )PCOPY SUPSMO &A
&STACK )SAVE
&STACK )OFF HOLD
EXEC APL
-DRP ERASE &FN DATA *
CP TERMINAL APL ON
&STACK )LOAD SUPSMO
&STACK )ERASE &A
&STACK )ERASE &1
&STACK )SAVE
&STACK )OFF HOLD
EXEC APL
-EX &TYPE YOU HAVE FINISHED
&EXIT 1000
-TELL &TYPE YOU HAVE ENTERED TOO MANY OR
&TYPE NOT ENOUGH ENTRIES ABOUT DATA FILE
```

```
&TYPE YOU NEED TO BEGIN AGAIN BY ENTERING
&TYPE
&TYPE    SUPSMO
&EXIT 100
&GOTO -EX
-ERROR &TYPE ABOVE ENTERED FILE DATA
&TYPE DOES NOT EXIST ON YOUR A-DISK
&TYPE CHECK YOUR FLIST AND
&TYPE THEN BEGIN AGAIN BY ENTERING
&TYPE
&TYPE            SUPSMO
&EXIT 101
-EXIT &TYPE YOU HAVE FORCED AN EXIT ON THIS SMOOTHING EXEC
&TYPE IF YOU WISH TO BEGIN AGAIN ENTER
&TYPE
&TYPE            SUPSMO
&EXIT 102
```

2. SUPSMO FORTRAN

The following file is SUPSMO FORTRAN which does the actual smoothing of a data set. The subroutines SUPSMU and SMOOTH of the following FORTRAN program were developed by Friedman and Stutzle [Ref. 9] as stated in Chapter I.

```
C
C   READ  SUPSMO EXEC  FILE BEFORE USING THIS FILE.
C
C***************************************************************************
C   THIS PROGRAM READS THE INPUT DATA, Y(N) VARIABLES FROM THE FILE      *
C   WATER DATA A1 AND THEN USES THE INTERNAL SUPER SMOOTHING SUBROUT.*
C   IN ORDER TO SMOOTH THE INPUT DATA.                                    *
C   THE SPANS CAN BE CHANGED BY ENTERING DESIRED SPANS ON THE VERY       *
C   LAST LINE OF THIS FILE.                                               *
C***************************************************************************
C
C
C INPUT:
C      N   : NUMBER OF OBSERVATIONS (X,Y - PAIRS)
C      X(N) : ORDERED ABSCISSA VALUES
C      Y(N) : CORRESPONDING ORDINATE (RESPONSE) VALUES
C      W(N) : WEIGHT FOR EACH (X,Y) OBSERVATION
C      IPER : PERIODIC VARIABLE FLAG
C        IPER=1 : X IS ORDERED INTERVAL VARIABLE
C        IPER=2 : X IS A PERIODIC VARIABLE WITH VALUES
C               IN THE RANGE (0.0, 1.0) AND PERIOD 1.0
C      SPAN : SMOOTHER SPAN (FRACTION OF OBSERVATIONS IN WINDOW).
C            SPAN=0.0 : AUTOMATIC (VARIABLE) SPAN SELECTION
C      ALPHA: CONTROLS HIGH FREQUENCY (SMALL SPAN) PENALTY
C            USED WITH AUTOMATIC SPAN SELECTION (BASE TONE CONTROL)
C            (ALPHA.LE.0.0 OR ALPHA.GT.10.0 : NO EFFECT)
C OUTPUT:
C      SMO(N): SMOOTHED ORDINATE (RESPONSE) VALUES
C SCRATCH:
C      SC(N,7): INTERNAL WORKING STORAGE
C NOTE:
C      FOR SMALL SAMPLES (N < 40) OR IF THERE ARE SUBSTANTIAL SERIAL
C      CORRELATIONS BETWEEN OBSERVATIONS CLOSE IN X - VALUE, THEN
C      A PRESPECIFIED FIXED SPAN SMOOTHER (SPAN > 0) SHOULD BE
C      USED.  REASONABLE SPAN VALUES ARE 0.3 TO 0.5.
C
C
         REAL*4 Y(5000),X(5000),SMO(5000),W(5000),SPAN,ALPHA,SC(5000,7)
         REAL*4 ACVR(5000),TPANS(3)
         INTEGER IR(5000),K,N,IPER,WEI,ODR
```

```
      DOUBLE PRECISION WT,FBO,FBW,XM,YM,TMP,VAR,CVAR,A,H,SY
      COMMON /CONSTS/ BIG,SML,EPS
      WRITE(5,1)
  1   FORMAT(1X,'ENTER THE NUMBER OF DATA POINTS TO BE SMOOTHED—
     *INTEGER VALUE')
      READ(6,*)N
 18   DO 19 I=1,N
 19   W(I)=1.
  9    WRITE(5,12)
 12    FORMAT(1X,'ARE THE INPUT DATA POINTS IN CHRONOLOGICAL
     *ORDER?',/,1X,'ENTER 0 FOR NO OR 1 FOR YES')
      READ(6,*)ODR
      IF(ODR.EQ.1)GO TO 13
      READ(4,*)(X(I),I=1,N)
      GO TO 14
 13   DO 15 I=1,N
 15   X(I)=FLOAT(I)
 14   CALL FRTCMS('CLRSCRN ')
      WRITE(5,5)
  5   FORMAT(1X,'ENTER 1.0 IF YOU DESIRE TO USE ONLY ONE SPAN VALUE',
     */,'ENTER 0.0 IF YOU WANT TO USE THREE SPAN VALUES')
      READ(6,*)SPAN
      CALL FRTCMS('CLRSCRN ')
      IF(SPAN.EQ.1.0)THEN
      WRITE(5,8)N
  8    FORMAT(1X,'ENTER THE SPAN VALUE TO BE USED',/,1X,'FRACTION OF',I5,
     *'I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0')
      READ(6,*)SPAN
      ALPHA = 0.0
      ELSE
      WRITE(5,2)N
  2    FORMAT(1X,'ENTER THE LOWEST SPAN VALUE:',/,1X,'FRACTION OF',
     *I5,' I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0')
      READ(6,*)TPANS(1)
      WRITE(5,3)N
  3    FORMAT(1X,'ENTER THE MIDDLE SPAN VALUE:',/,1X,'FRACTION OF'
     *,I5,'I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0')
      READ(6,*)TPANS(2)
      WRITE(5,4)N
  4    FORMAT(1X,'ENTER THE HIGHEST SPAN VALUE:',/,1X,'FRACTION OF'
     *,I5,' I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0')
      READ(6,*)TPANS(3)
      CALL FRTCMS('CLRSCRN ')
 11   WRITE(5,16)
 16    FORMAT(1X,'IF ONE OF THE SPAN VALUES IS SMALL',/,
     *'I.E. RESULTS IN A SMALL WINDOW SIZE (10 OR LESS)',/,
     *'YOU MAY WISH TO ADJUST THE SMOOTH CURVE ROBUSTNESS',/,
     *'BY ENTERING A REAL NUMBER GT 0.0 BUT LT 10.0',/,
     *'FOR NO ROBUST ADJUSTMENT ENTER 0.0',/,
     *'OR COMPLETE ROBUST ADJUSTMENT ENTER 10.0',/,
     *'ENTER YOUR CHOICE')
      READ(6,*)ALPHA
      CALL FRTCMS('CLRSCRN ')
      ENDIF
      WRITE(6,20)
 20   FORMAT(1X,'****PLEASE WAIT SMOOTHING PROGRAM NOW RUNNING*****')
      READ(7,*)(Y(I),I=1,N)
      IF(ODR.EQ.1)GO TO 17
      CALL SORTER(X,W,Y,N)
 17   IPER=1
      IF(X(N).EQ.1.0)IPER=2
  7   CALL SUPSMU(N,X,Y,W,IPER,SPAN,ALPHA,SMO,SC,TPANS)
      WRITE(8,10)(SMO(I),I=1,N)
 10   FORMAT(2X,5(F12.6,2X))
      STOP
      END
C*********************************************************************
C
C*********************************************************************
      SUBROUTINE SUPSMU(N,X,Y,W,IPER,SPAN,ALPHA,SMO,SC,TPANS)
      DIMENSION X(N),Y(N),W(N),SMO(N),SC(N,7),TPANS(3)
      COMMON /CONSTS/ BIG,SML,EPS
      IF (X(N).GT.X(1))GO TO 30
      SY=0.0
      SW=SY
      DO 10 J=1,N
           SY=SY+W(J)*Y(J)
           SW=SW+W(J)
10    CONTINUE
      A=SY/SW
      DO 20 J=1,N
           SMO(J)=A
20    CONTINUE
      RETURN
 30   I=N/4
      J=3*I
      SCALE=X(J)-X(I)
 40   IF(SCALE.GT.0.0)GO TO 50
      IF(J.LT.N)J=J+1
      IF(I.GT.1)I=I-1
      SCALE=X(J)-X(I)
      GO TO 40
 50   VSMLSQ=(EPS*SCALE)**2
      JPER=IPER
      IF(IPER.EQ.2.AND.(X(1).LT.0.0.OR.X(N).GT.1.0))JPER=1
      IF(JPER.LT.1.OR.JPER.GT.2)JPER=1
      IF(SPAN.LE.0.0)GO TO 60
      CALL SMOOTH(N,X,Y,W,SPAN,JPER,VSMLSQ,SMO,SC)
      RETURN
 60   DO 70 I=1,3
          CALL SMOOTH(N,X,Y,W,TPANS(I),JPER,VSMLSQ,SC(1,2*I-1),SC(1,7))
          CALL SMOOTH(N,X,SC(1,7),W,TPANS(2),-JPER,VSMLSQ,SC(1,2*I),H)
 70   CONTINUE
      DO 90 J=1,N
          RESMIN=BIG
          DO 80 I=1,3
                 IF(SC(J,2*I).GE.RESMIN)GO TO 80
                 RESMIN=S(J,2*I)
                 SC(J,7)=TPANS(I)
 80       CONTINUE
               IF(ALPHA.GT.0.0.AND.ALPHA.LE.10.0.AND.RESMIN.LT.SC(J,6))SC(J,7)
     *    =SC(J,7)+(TPANS(3)-SC(J,7))*AMAX1(SML,RESMIN/SC(J,6))**(10.0-
     *    ALPHA)
 90   CONTINUE
      CALL SMOOTH(N,X,SC(1,7),W,TPANS(2),-JPER,VSMLSQ,SC(1,2),H)
      DO 110 J=1,N
           IF(SC(J,2).LE.TPANS(1))SC(J,2)=TPANS(1)
           IF(SC(J,2).GE.TPANS(3))SC(J,2)=TPANS(3)
           F=SC(J,2)-TPANS(2)
           IF(F.GE.0.0)GO TO 100
           F=-F/(TPANS(2)-TPANS(1))
           SC(J,4)=(1.0-F)*SC(J,3)+F*SC(J,1)
           GO TO 110
100        F=F/(TPANS(3)-TPANS(2))
           SC(J,4)=(1.0-F)*SC(J,3)+F*SC(J,5)
110   CONTINUE
      CALL SMOOTH(N,X,SC(1,4),W,TPANS(1),-JPER,VSMLSQ,SMO,H)
      RETURN
      END
C*****************************************************************
C
C*****************************************************************
      SUBROUTINE SMOOTH(N,X,Y,W,SPAN,IPER,VSMLSQ,SMO,ACVR)
      DIMENSION X(N),Y(N),W(N),SMO(N),ACVR(N)
      INTEGER IN,OUT
      DOUBLE PRECISION WT,FBO,FBW,XM,YM,TMP,VAR,CVAR,A,H,SY
      XM=0.0
      YM=XM
      VAR=YM
      CVAR=VAR
      FBW=CVAR
      JPER=IABS(IPER)
      IBW=0.5*SPAN*N+0.5
      IF(IBW.LT.2)IBW=2
      IT=2*IBW+1
      DO 20 I=1,IT
          J=I
          IF(JPER.EQ.2)J=I-IBW-1
          XTI=X(J)
          IF(J.GE.1)GO TO 10
          J=N+J
          XTI=X(J)-1.0
10        WT=W(J)
          FBO=FBW
          FBW=FBW+WT
          XM=(FBO*XM+WT*XTI)/FBW
          YM=(FBO*YM+WT*Y(J))/FBW
          TMP=0.0
          IF(FBO.GT.0.0)TMP=FBW*WT*(XTI-XM)/FBO
          VAR=VAR+TMP*(XTI-XM)
          CVAR=CVAR+TMP*(Y(J)-YM)
20    CONTINUE
      DO 70 J=1,N
          OUT=J-IBW-1
          IN=J+IBW
          IF((JPER.NE.2).AND.(OUT.LT.1.OR.IN.GT.N))GO TO 60
          IF(OUT.GE.1)GO TO 30
          OUT=N+OUT
          XTO=X(OUT)-1.0
          XTI=X(IN)
          GO TO 50
30        IF(IN.LE.N)GO TO 40
          IN=IN-N
          XTI=X(IN)+1.0
          XTO=X(OUT)
          GO TO 50
40        XTO=X(OUT)
          XTI=X(IN)
50        WT=W(OUT)
          FBO=FBW
          FBW=FBW-WT
          TMP=0.0
          IF(FBW.GT.0.0)TMP=FBO*WT*(XTO-XM)/FBW
          VAR=VAR-TMP*(XTO-XM)
          CVAR=CVAR-TMP*(Y(OUT)-YM)
          XM=(FBO*XM-WT*XTO)/FBW
          YM=(FBO*YM-WT*Y(OUT))/FBW
          WT=W(IN)
          FBO=FBW
          FBW=FBW+WT
          XM=(FBO*XM+WT*XTI)/FBW
          YM=(FBO*YM+WT*Y(IN))/FBW
          TMP=0.0
          IF(FBO.GT.0.0)TMP=FBW*WT*(XTI-XM)/FBO
          VAR=VAR+TMP*(XTI-XM)
          CVAR=CVAR+TMP*(Y(IN)-YM)
60        A=0.0
          IF(VAR.GT.VSMLSQ)A=CVAR/VAR
          SMO(J)=A*(X(J)-XM)+YM
          IF(IPER.LE.0)GO TO 70
          H=1.0/FBW
          IF(VAR.GT.VSMLSQ)H=H+(X(J)-XM)**2/VAR
          ACVR(J)=ABS(Y(J)-SMO(J))/(1.0-W(J)*H)
70    CONTINUE
      J=1
80    JO=J
      SY=SMO(J)*W(J)
      FBW=W(J)
      IF(J.GE.N)GO TO 100
90    IF(X(J+1).GT.X(J))GO TO 100
```

```
          J=J+1
          SY=SY+W(J)*SMO(J)
          FBW=FBW+W(J)
          IF(J.LT.N)GO TO 90
  100     IF(J.LE.JO)GO TO 120
          SY=SY/FBW
          DO 110 I=JO,J
               SMO(I)=SY
  110     CONTINUE
  120     J=J+1
          IF(J.LE.N)GO TO 80
          RETURN
          END
C*****************************************************************
C
C*****************************************************************
          SUBROUTINE SORTER(X,W,Y,N)
          REAL*4 X(N),W(N),Y(N),D(5000)
          INTEGER N,KEY(5000)
               DO 5 I=1,N
    5          KEY(I)=I
               CALL SHSORT(X,KEY,N)
               DO 1 I=1,N
    1          D(I)=W(I)
               DO 2 I=1,N
                    J=KEY(I)
    2          W(I)=D(J)
               DO 3 I=1,N
    3          D(I)=Y(I)
               DO 4 I=1,N
                    J=KEY(I)
    4          Y(I)=D(J)
          RETURN
          END
C*****************************************************************
C
C*****************************************************************
          BLOCK DATA
C
C---------------------------------------------
C
C THIS SETS THE COMPILE TIME (DEFAULT) VALUES FOR VARIOUS
C INTERNAL PARAMETERS:
C   BIG    : A LARGE REPRESENTATIVE FLOATING POINT NUMBER
C   SMALL  : A SMALL NUMBER. SHOULD BE SET SO THAT (SML)**(10.0)
C            DOES NOT CAUSE FLOATING POINT UNDERFLOW
C   EPS    : USED TO NUMERICALLY STABILIZE SLOPE CALCULATIONS FOR
C            RUNNING LINEAR FITS
C   THESE PARAMETER VALUES CAN BE CHANGED BY DECLARING THE RELEVANT
C   LABELED COMMON IN THE MAIN PROGRAM AND RESETTING THEM WITH
C   EXECUTABLE STATEMENTS.
C
C---------------------------------------------
C
          COMMON /CONSTS/ BIG,SML,EPS
          DATA BIG,SML,EPS /1.0E20,1.0E-7,1.0E-3/
          END
```

3. *SUPSMO VSAPLWS*

The following two APL functions are used in conjuction with the two files listed above. They were developed by the author of this thesis, and are the main APL functions within the APL workspace SUPSMO.

The first APL function links the user of the smoothing program with GRAFSTAT and gives a user familiar with GRAFSTAT the opportunity to proceed into GRAFSTAT where a greater variety of graphic functions are available.

```
  ∇ ST RT;C
[1]    DUM←CMS 'CLRSCRN'
[2]    'THE ENTIRE DATA FILE THAT YOU WANTED SMOOTHED HAS
BEEN TRANSFERRED'
[3]    'TO THIS WORKSPACE SO THAT YOU MAY BE ABLE TO PLOT
BOTH'
[4]    'THE SMOOTHED AND UNSMOOTHED DATA.'
[5]    'THE UNSMOOTHED DATA IS IN THE VARIABLE WITH THE SAME
NAME AS'
[6]    'THE DATA FILE THAT YOU HAVE YOUR INPUT DATA IN.'
[7]    ' '
[8]    ' '
[9]    'DO YOU WISH TO GO INTO APL OR CONTINUE?'
[10]   'ENTER 0 FOR APL OR 1 FOR CONTINUE'
[11]   C←⎕
[12]   →13+C×8
[13]   'YOU WILL BE SENT TO APL AFTER YOU HAVE READ THIS
IMPORTANT TEXT.'
[14]   '***AFTER YOU HAVE FINISHED WORKING IN APL AND WISH TO
PLOT THE DATA.'
[15]   'ENTER PLOTER*******NOTICE THAT PLOTER HAS ONLY ONE T'
[16]   ' '
[17]   ' '
[18]   'NOW ENTER 0 AGAIN'
[19]   C←⎕
[20]   →C
[21]   PLOTER
  ∇
```

The next APL function creates the APL variables to be used in the GRAFSTAT 'PLOT' screen. This plotting option is made available to the user through the above APL function. The user can use this APL function to do the plotting or use the GRAFSTAT graphics functions. A user need not fully understand how to use the GRAFSTAT plot screen in order to use this function. Several examples are shown with each requested entry so that the user can see what the entry should look like.

```
  ∇
PLOTER;  ;CO;DUM;P;SY;TI;TL;TP;XL;XO;XS;XT;XY;XV;YL;YO;YS;YT;YV;YY
[1]    DUM←CMS 'CLRSCRN'
[2]    'YOU HAVE ACTIVATED THE PLOTTING FUNCTION'
[3]    'IT IS ASSUMED THAT THE USER IS FAMILIAR WITH THE
GRAFSTAT PLOT FUNC.'
[4]    'AND THE AXIS CONTROL FUNCTION'
[5]    'IF YOU RECEIVE (MAKE) AN ERROR MESSAGE DO THE
FOLLOWING'
[6]    '1.  ENSURE THAT VM READ IS DISPLAYED IN LOWER RIGHT
CORNER OF SCREEN'
[7]    '2.  PRESS THE ENTER KEY'
[8]    '3.  ENTER        PAGE'
[9]    'TO UNDERSCORE A LETTER HOLD THE APL/ALT KEY DOWN AND
PRESS THE LETTER'
[10]   'THE PLOTTING FUNCTION WILL RESTART AT THE BEGINNING'
```

```
[11]   'THE  PLOTTING  FUNCTION  CAN  BE  EXITED  AT  ANY  INPUT
POINT BY ENTERING'
[12]   '                 →'
[13]   'AT ANYTIME THAT YOU EXIT THE PLOTTING FUNCTION'
[14]   'YOU WILL BE IN THE GRAFSTAT WORKSPACE'
[15]   'IF YOU WISH TO RETURN TO CMS ENTER'
[16]   '            )OFF HOLD'
[17]   '     '
[18]   '     '
[19]  LB:'ENTER  X  VARIABLE(S)  (ENCLOSED  IN  QUOTES),  IF
ENTERING MORE THAN ONE VARIABLE'
[20]   'SEPARATE VARIABLES WITH SEMICOLON AND USE QUOTES'
[21]   'E.G. ''X'' OR ''X1;X2''  '
[22]   XV←⍞
[23]   DUM←CMS 'CLRSCRN'
[24]   'ENTER Y VARIABLE(S) (ENCLOSED IN QUOTES AND MUST BE
OF SAME LENGTH AS X)'
[25]   'IF  ENTERING  MORE  THAN  ONE  VARIABLE,  SEPARATE  WITH
SEMICOLON'
[26]   'AND REMEMBER TO USE QUOTES ENCLOSING ENTIRE STRING'
[27]   'E.G. ''Y'' OR ''Y1;Y2''   '
[28]   YV←⍞
[29]   'ENTER  A  VECTOR  INDICATING  TYPE(S)  OF  PLOT;  0≡SYM
ONLY; 1≡LINE ONLY'
[30]   'E.G. 0 OR 1 OR 0 1 OR 0 0 OR 1 0 OR 1 1'
[31]   TP←⍞
[32]   →((×/TP)>0)/L1
[33]   'ENTER  TYPE  OF  SYMBOL  CORRESPONDING  TO  EACH  SYMBOLS
ONLY PLOT (IN QUOTES)'
[34]   'E.G. ''.'' OR ''.*'' YOU CAN USE .*+×'
[35]   SY←⍞
[36]   →((+/TP)≡0)/LP
[37]  L1:'ENTER A VECTOR INDICATING TYPE(S) OF LINES; 1≡SOLID
LINE; 3≡DASH LINE'
[38]   'E.G. 1 OR 3 OR 1 3 OR ANY OTHER COMBINATION OR LINE
TYPES IN GRAFSTAT'
[39]   TL←⍞
[40]  LP:TL←1
[41]   →((ρTP)≥1)/L2
[42]   SY←'.'
[43]  L2:DUM←CMS 'CLRSCRN'
[44]   'ENTER  SCALE  OF  X-AXIS  (IN  QUOTES)  OR  P  (IN  QUOTES)
FOR PREVIOUS SCALE'
[45]   'E.G. ''LIN'' OR ''LIN XMIN XMAX'' OR ''P''  '
[46]   XS←⍞
[47]   'ENTER  SCALE  OF  Y-AXIS  (IN  QUOTES)  OR  P  (IN  QUOTES)
FOR PREVIOUS SCALE'
[48]   'E.G. ''LIN'' OR ''LIN YMIN YMAX'' OR ''P''   '
[49]   YS←⍞
[50]   'ENTER THE PLOT HEADER (IN QUOTES) OR EMPTY QUOTES'
[51]   'E.G. ''TITLE'' OR '' ''  '
[52]   TI←⍞
[53]   DUM←CMS 'CLRSCRN'
[54]   'ENTER X-AXIS LABEL (IN QUOTES) OR '
[55]   'A  PAIR  OF  EMPTY  QUOTES  FOR  NO  LABEL  OR  TO  USE  AXIS
CONTROL'
[56]   'E.G. ''LABEL'' OR '' ''  '
[57]   XL←⍞
[58]   'ENTER Y-AXIS LABEL (IN QUOTES) OR'
[59]   'A  PAIR  OF  EMPTY  QUOTES  FOR  NO  LABEL  OR  TO  USE  AXIS
CONTROL'
[60]   'E.G. ''LABLE'' OR '' ''  '
[61]   YL←⍞
[62]   'DO YOU WANT TO RUN THIS PAGE?'
[63]   'ENTER 0 FOR NO OR 1 FOR YES'
[64]   CO←⍞
[65]   DUM←CMS 'CLRSCRN'
[66]   →L3+4×(CO≡0)
[67]  L3:P← 1 1 1
[68]      ← 0 1 0 0
```

```
[69]  'PLEASE WAIT RUNNING PAGE'
[70]  RUN PAGESAM
[71]  'DO YOU WANT TO EXIT THIS FUNCTION?'
[72]  'ENTER 0 FOR NO OR 1 FOR YES'
[73]  CO←⎕
[74]  →(CO≡1)/LE
[75]  'DO YOU WANT TO RESTART THIS FUNCTION?'
[76]  'ENTER 0 FOR NO OR 1 FOR YES'
[77]  CO←⎕
[78]  DUM←CMS 'CLRSCRN'
[79]  →(CO≡1)/LB
[80]  'THE ONLY THING LEFT TO DO IS THE AXIS CONTROL'
[81] L6:'WITH THE PARTIAL PLOT THAT YOU HAVE JUST FINISHED
CONSTRUCTING'
[82]  'ENTER  A 3 ELEMENT VECTOR FOR PARTIAL PLOT'
[83]  '1ST ELEMENT, 1(0): LINES AND SYMBOLS ARE (NOT) SHOWN
ON SCREEN'
[84]  '2ND ELEMENT, 1(0): HEADER AND AXES ARE (NOT) SHOWN ON
SCREEN'
[85]  '3RD ELEMENT, 1(0): AXES, GRIDS, AND GRID LINES ARE
(NOT) SHOWN'
[86]  'E.G. 1 1 0 WILL SHOW EVERYTHING ON GRAPH EXCEPT AXES
AND GRID LINES'
[87]  P←⎕
[88]  'ENTER A 4 ELEMENT VECTOR FOR AXES AND GRID CONTROL'
[89]  '1ST ELEMENT, X-AXIS: 0 ≡ BOTTOM, 2 ≡ TOP, OR 20 ≡ AT
Y≡0'
[90]  '2ND ELEMENT, Y-AXIS: 1 ≡ LEFT, 3 ≡ RIGHT, OR 21 ≡ AT
X≡0'
[91]   '3RD  ELEMENT,  VERTICAL  GRID  LINES:  0≡NO  GRID,
1≡DOTTED, OR 2≡SOLID'
[92]   '4TH  ELEMENT,  HORIZON.  GRID  LINES:  0≡NO  GRID,
1≡DOTTED, OR 2≡SOLID'
[93]  'E.G. 2 1 2 2 WILL DISPLAY AXIS AT TOP AND LEFT AND
SOLID GRID LINES'
[94]      ←⎕
[95] L8:'PLEASE WAIT RUNNING PAGE'
[96]  RUN PAGESAM
[97] LA:DUM←CMS 'CLRSCRN'
[98]  'ENTER X-AXIS TIC MARKS LOCATION VECTOR'
[99]  'OR ENTER 0 FOR STANDARD TIC MARKS'
[100]  'OR ENTER 1 FOR NO TIC MARKS'
[101]  'E.G. 1 5 11 OR A VECTOR NAME OR 0 OR 1'
[102]  '1 5 11 WILL SHOW TIC MARKS AT X≡1, X≡5, AND X≡11'
[103]  XT←⎕
[104]  'ENTER X-AXIS SYMBOLS (IN QUOTES)'
[105]  'OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS'
[106]  'OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS'
[107]  'E.G. ''1970;1971'' OR A VECTOR NAME OR 0 OR 1'
[108]  XY←⎕
[109]  'ENTER X-AXIS SYMBOLS LOCATIONS VECTOR'
[110]  'OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO
SYMBOLS'
[111]  'E.G. 6 18 OR A VECTOR NAME OR 0'
[112]  '6 18 WILL SHOW 1970 AT X≡6 AND 1971 AT X≡18'
[113]  XO←⎕
[114]  DUM←CMS 'CLRSCRN'
[115]  'ENTER Y-AXIS TIC MARKS LOCATION VECTOR'
[116]  'OR ENTER 0 FOR STANDARD TIC MARKS'
[117]  'OR ENTER 1 FOR NO TIC MARKS'
[118]  'E.G. ¯1 0 1 OR A VECTOR NAME OR 0 OR 1'
[119]  '¯1 0 1 WILL SHOW TIC MARKS AT Y≡¯1, Y≡0, AND Y≡1'
[120]  YT←⎕
[121]  'ENTER Y-AXIS SYMBOLS (IN QUOTES)'
[122]  'OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS'
[123]  'OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS'
[124]  'E.G. ''LO MID HI'' OR VECTOR NAME OR 0 OR 1'
[125]  YY←⎕
[126]  'ENTER Y-AXIS SYMBOLS LOCATIONS VECTOR'
```

```
[127]  'OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO
SYMBOLS'
[128]  'I.E. ¯1 0 1 OR VECTOR NAME OR 0'
[129]  '¯1 0 1 WILL SHOW LO AT Y≡¯1, MID AT Y≡0, HI AT Y≡1'
[130]  YO←⎕
[131]  'THESE AXIS CONTROL ENTRIES WILL NOW BE RUN'
[132]  RUN PAGEAX
[133]  'DO YOU WANT TO RERUN THE PLOT INPUTS YOU ENTERED'
[134]  'BEFORE RUNNING THIS AXIS CONTROL FUNCTION?'
[135]  'ENTER 0 FOR NO OR 1 FOR YES'
[136]  CO←⎕
[137]  →(CO≡1)/L6
[138]  'DO YOU WANT TO DO ANOTHER AXIS CONTROL PAGE?'
[139]  'ENTER 0 FOR NO OR 1 FOR YES'
[140]  CO←⎕
[141]  →(CO≡1)/L8
[142]  'DO YOU WANT TO RESTART THE FUNCTION?'
[143] LE:'IF YOU DO NOT YOU WILL EXIT THIS FUNCTION'
[144]  'IF YOU EXIT THIS FUNCTION AND WANT TO RETAIN THIS
WORK'
[145]  'USE THE KEEP FUNCTION AND THEN YOU CAN RETURN TO
CMS'
[146]  'BY ENTERING )OFF HOLD'
[147]  'IF YOU WANT TO RETURN TO CMS, SIMPLY ENTER )OFF HOLD
AFTER EXIT'
[148]  'ENTER 0 FOR EXIT OR 1 FOR RESTART'
[149]  CO←⎕
[150]  →(CO≡1)/LB
[151]  ⍳0
   ∇
```

APPENDIX B

SPLIT LINEAR FIT PROGRAM

1. *SPTLIN EXEC*

The following file is the exec file, SPTLIN EXEC, which activates and runs the Split Linear Fit smoothing program. Chapter VI contains instructions on how to use this smoothing program.

```
&TRACE
SET BLIP *
GLOBAL  TXTLIB  VLNKMLIB  VALTLIB  VFORTLIB  IMSLSP  NONIMSL
CMSLIB
CLRSCRN
&TYPE YOU HAVE INITIATED AN ALGORITHM
&TYPE TO SMOOTH A SET OF DATA USING
&TYPE 'SMOOTHING WITH SPLIT LINEAR FITS'
&TYPE DEVELOPED BY MCDONALD AND OWEN OF
&TYPE STANFORD UNIVERSITY DEPT. OF STATISTICS
-STRT &TYPE *******************************************
&TYPE IN ORDER TO USE THIS ALGORITHM USE A 2M MACHINE**
&TYPE *************************************************
&TYPE
&TYPE
&TYPE DO YOU WISH TO CONTINUE?
&TYPE ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:
&READ VAR &CONT
CLRSCRN
&IF &CONT NE Y &GOTO -EXIT
&TYPE IN ORDER TO USE THIS ALGORITHM YOU MUST
&TYPE HAVE ON HAND THE FOLLOWING:
&TYPE
&TYPE 1.  FILENAME OF DATA FILE (FILETYPE DATA)
&TYPE     WITH DATA TO BE SMOOTHED.
&TYPE
&TYPE 2.  IF DATA POINTS ARE NOT IN CHRONOLOGICAL ORDER,
&TYPE     YOU NEED TO HAVE A FILE (FILETYPE ORDER)
&TYPE     WITH INDICES CORRESPONDING TO DATA POINTS
&TYPE     INDICATING THE ORDER OF THE DATA POINTS.
&TYPE
&TYPE 3.  FILENAME OF DATA FILE WHERE SMOOTHED OUTPUT
&TYPE     WILL BE WRITTEN OR IF YOU WANT
&TYPE     TO WRITE OUTPUT INTO APL HAVE ON HAND
&TYPE     THE APL VARIABLE AND WORKSPACE NAMES
&TYPE     THAT WILL STORE THE OUTPUT.
&TYPE
&TYPE 4.  THE NUMBER OF WINDOW SIZES
&TYPE     AND THE VALUE OF THE WINDOW SIZES
&TYPE     THAT YOU WANT TO ATTEMPT
&TYPE     ON THE SMOOTHING OF THE DATA.
&TYPE
&TYPE 5.  THE MINIMUM WINDOW SIZE THAT
&TYPE     CAN BE ATTEMPTED BY THE ALGORITHM.
&TYPE
&TYPE DO YOU WISH TO CONTINUE?
&TYPE ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:
&READ VAR &CONT
CLRSCRN
&IF &CONT NE Y &GOTO -EXIT
&TYPE CAN YOU ACCESS 2M OF STORAGE ON THIS DISK?
```

```
&TYPE ENTER Y FOR YES OR N FOR NO:
&READ VAR &CONT
&IF &CONT EQ N &GOTO -STRT
&IF &CONT NE Y &GOTO -EXIT
&TYPE ENTER FILENAME OF FILE WHICH
&TYPE CONTAINS THE DATA TO BE SMOOTHED:
&READ ARGS
&IF &N ≡ 0 &GOTO -TELL
&IF &N > 1 &GOTO -TELL
STATE &1 DATA A1
&IF &RC NE 0 &GOTO -ERROR
CLRSCRN
-ORDR CLRSCRN
&TYPE ARE DATA POINTS TO BE SMOOTHED
7type IN CHRONOLOGICAL ORDER?
&TYPE ENTER Y FOR YES OR N FOR NO:
&READ VAR &CONT
&IF &CONT EQ Y &GOTO -GO
&TYPE THE DATA POINTS ARE NOT
&TYPE IN CHRONOLOGICAL ORDER?
&TYPE ENTER FILENAME OF FILE (FILETYPE ORDER)
&TYPE THAT CONTAINS ORDER INDICES
&READ VAR &ORD
STATE &ORD ORDER A1
&IF &RC NE 0 &GOTO -ERROR
-GO CLRSCRN
&TYPE THE DATA YOU WANT TO SMOOTH
&TYPE IS IN     &1 DATA
&TYPE WHERE DO YOU WANT TO WRITE THE SMOOTHED OUTPUT?
&TYPE CMS OR APL
&TYPE OR YOU CAN PLOT THE SMOOTHED OUTPUT
&TYPE SINCE YOU ARE LOGGED ON A
&TYPE TERMINAL THAT CAN ACCESS GRAFSTAT,
&TYPE BUT THE OUTPUT MUST BE STORED
&TYPE IN AN APL VARIABLE.
&TYPE ENTER APL OR CMS
&READ VARS &PLA
&IF &PLA EQ APL &GOTO -AP1
&TYPE
CLRSCRN
&TYPE THE SMOOTHED OUTPUT WILL BE WRITTEN
&TYPE INTO A CMS FILE (FILETYPE DATA)
&TYPE ENTER ONLY THE FILENAME.
&READ VAR &FN
&TYPE THE SMOOTHED OUTPUT WILL BE WRITTEN
&TYPE INTO THE FILE &FN DATA
&GOTO -COM
-AP1 &TYPE
&TYPE ENTER THE NAME OF THE
&TYPE APL VARIABLE TO HOLD THE OUTPUT
&READ VAR &A
&TYPE DO YOU WANT TO PLOT THE OUTPUT?
&TYPE ENTER Y FOR YES OR N FOR NO:
&READ VAR &GRF
&IF &GRF EQ Y &GOTO -PLOT
&TYPE ENTER THE NAME OF THE APL WORKSPACE
&TYPE THAT WILL CONTAIN &A
&READ VAR &WKS
&TYPE IS &WKS AN EXISTING WORKSPACE OR A NEW WORKSPACE?
&TYPE ENTER O FOR EXISTING OR N FOR NEW:
&READ VAR &AGE
-PLOT &FN ≡ TE
-COM &TYPE
&TYPE PLEASE READ THE FOLLOWING INSTRUCTIONS VERY CAREFULLY
&TYPE ARE YOU READY TO START THE SMOOTHING PROGRAM?
&TYPE ENTER Y FOR YES ANY OTHER KEY TO EXIT
&READ VARS &O
CLRSCRN
&IF &O NE Y &GOTO -EXIT
&TYPE PLEASE WAIT THE SMOOTHING PROGRAM IS BEING COMPILED
```

```
FI 04 CLEAR
FI 05 CLEAR
FI 06 CLEAR
FI 07 CLEAR
FI 08 CLEAR
FI 09 CLEAR
FORTVS SPTLIN(LVL (77) SDUMP
FIL 04 DISK &ORD ORDER
FIL 07 DISK &1 DATA
FIL 08 DISK &FN DATA (RECFM FBA LRECL 80 BLKSIZE 800)
CLRSCRN
&TYPE PLEASE WAIT SMOOTHING PROGRAM IS BEING LOADED
LOAD SPTLIN (START
CLRSCRN
ERASE SPTLIN LISTING
ERASE SPTLIN TEXT
ERASE LOAD MAP
&IF &PLA EQ CMS &GOTO -EX
&IF &GRF EQ N &GOTO -NGRF
CP TERMINAL APL ON
&STACK )LOAD SPTLIN
&STACK &A ←CMSREAD
&STACK &FN
&STACK DATA
&STACK N
&STACK &A ← ,&A
&STACK &1 ←CMSREAD
&STACK &1
&STACK DATA
&STACK N
&STACK &1 ← ,&1
&STACK )SAVE
&TYPE ****PLEASE WAIT, LINKING TO GRAFSTAT**************
&STACK )LOAD GRAFSTAT
&STACK DUM ←CMS 'CLRSCRN'
&STACK )PCOPY SPTLIN
&STACK ST RT
EXEC APLGS
&GOTO -DRP
-NGRF CP TERMINAL APL ON
&STACK )LOAD SPTLIN
&STACK &A ←CMSREAD
&STACK &FN
&STACK DATA
&STACK N
&STACK &A ← ,&A
&STACK )SAVE
&STACK )CLEAR
&IF &AGE EQ O &STACK )LOAD &WKS
&IF &AGE EQ N &STACK )WSID &WKS
&STACK )PCOPY SPTLIN &A
&STACK )SAVE
&STACK )OFF HOLD
EXEC APL
-DRP ERASE &FN DATA *
CP TERMINAL APL ON
&STACK )LOAD SPTLIN
&STACK )ERASE &A
&STACK )ERASE &1
&STACK )SAVE
&STACK )OFF HOLD
EXEC APL
-EX &TYPE YOU HAVE FINISHED.
&EXIT 1000
-TELL &TYPE YOU HAVE ENTERED TOO MANY OR
&TYPE NOT ENOUGH ENTRIES ABOUT DATA FILE
&TYPE YOU NEED TO BEGIN AGAIN
&TYPE ENTER:     SPTLIN
&EXIT 100
-ERROR &TYPE ABOVE ENTERED FILE DATA
```

```
&TYPE DOES NOT EXIST ON YOUR A-DISK
&TYPE CHECK YOUR FLIST AND THEN BEGIN AGAIN BY ENTERING
&TYPE       SPTLIN
&EXIT 101
-EXIT &TYPE YOU HAVE FORCED AN EXIT ON THIS SMOOTHING EXEC
&TYPE IF YOU WISH TO BEGIN AGAIN ENTER
&TYPE        SPTLIN
&EXIT 102
```

2. *SPTLIN FORTRAN*

The following file is SPTLIN FORTRAN which does the actual smoothing of a data set. The subroutines used in this program were developed by McDonald and Owen [Ref. 10] as stated in Chapter I. These subroutines were originally written in the C computer language. The author of this thesis translated the C language subroutines and combined them into an interactive FORTRAN program which follows.

```
      DOUBLE PRECISION WEIGHT,Y(1000),X(1000),W(1000),TRY(1000,8,9)
      DOUBLE PRECISION TSMO(1000),SMOOTH(1000)
      REAL INFIN, CEPS, MISVAL, RESCAL, WTPOW
      INTEGER CMXOBS, CMXTRY, MNWNSZ, BASE, TRYSPN(10), NOBS, NTRYS,
     *NOTMIS
      COMMON /CONSTS/ INFIN,CEPS;MISVAL,RESCAL,WTPOW
      WRITE(5,2)

    2 FORMAT(1X,'ENTER THE NUMBER OF DATA POINTS TO BE SMOOTHED
     *---INTEGER VALUE:',/)
      READ(6,*)NOBS
 13   DO 15 I = 1,NOBS
 15   W(I) = 1.
 14   WRITE(5,16)
 16   FORMAT(1X,'ARE THE INPUT DATA POINTS IN CHRONOLOGICAL ORDER?'
     *,/,1X,'ENTER 0 FOR NO OR 1 FOR YES',/)
      READ(6,*) ODR
      IF(ODR.EQ.1)GO TO 17
      READ(4,*)(X(I),I = 1,NOBS)
      GO TO 18
 17   DO 19 I = 1,NOBS
 19   X(I) = FLOAT(I)
 18   CALL FRTCMS('CLRSCRN ')
      WRITE(5,5)
    5 FORMAT(1X,'ENTER THE NUMBER WINDOWS TO BE USED
     *------INTEGER VALUE ')
      READ(6,*)NTRYS
      CALL FRTCMS('CLRSCRN ')
      WRITE(5,10)
 10   FORMAT(1X,'NEXT ENTER THE WINDOW SIZES IN INCREASING ORDER
      *--------INTEGER VALUES',/)
      DO 8 I = 1,NTRYS
         WRITE(5,9)I
  9      FORMAT(1X,'ENTER WINDOW SIZE NUMBER',I4,/)
         READ(6,*)TRYSPN(I)
         CALL FRTCMS('CLRSCRN ')
  8   CONTINUE
      WRITE(5,11)
 11   FORMAT(1X,'ENTER VALUE OF THE MINIMUM WINDOW SIZE
      *---INTEGER',/)
      READ(6,*)MNWNSZ
      CALL FRTCMS('CLRSCRN ')
      WRITE(5,20)
 20   FORMAT(1X,'*****PLEASE WAIT PROGRAM NOW RUNNING********')
      RESCAL = 1.0
```

```
         WTPOW = 2.0
         READ(7,*)(Y(I),I = 1, NOBS)
         IF(ODR.EQ.1)GO TO 21
         CALL SORTER(X,W,Y,N)
  21     CONTINUE
         DO 12 I=1,NTRYS
         DO 12 J=1,NOBS
               TRY(J,I,1)=0.0
               TRY(J,I,2)=0.0
               TRY(J,I,3)=0.0
               TRY(J,I,4)=0.0
               TRY(J,I,5)=0.0
               TRY(J,I,6)=0.0
               TRY(J,I,7)=0.0
               TRY(J,I,8)=0.0
               TRY(J,I,9)=0.0
   12    CONTINUE
         DO 6 I = 1, NTRYS
               CALL RUNLRC(NOBS, X, Y, W, NTRYS, MNWNSZ, TRYSPN,I,TRY)
     6   CONTINUE
         CALL COMPWT(NOBS, NTRYS, TRY)
         CALL COMTRY(NOBS, NTRYS, TSMO, TRY)
         DO 7 I = 1, NTRYS
               CALL RUNLRC(NOBS, X, TSMO, W, NTRYS,MNWNSZ, TRYSPN,I, TRY)
     7   CONTINUE
               CALL COMPWT(NOBS, NTRYS,TRY)
               CALL COMTRY(NOBS, NTRYS, SMOOTH, TRY)
         WRITE(8,4) (SMOOTH(I),I = 1, NOBS)
     4   FORMAT(2X,5(F12.6,2X))
         STOP
         END
C*********************************************************
C
C*********************************************************
         SUBROUTINE COMPWT(NOBS, NTRYS,TRY)
            DOUBLE PRECISION LAMBDA,TEMP,MIN,TRY(NOBS,NTRYS,9)
            REAL INFIN,CEPS,MISVAL,RESCAL,WTPOW
            INTEGER NOBS, NTRYS, NTMS, A
            COMMON /CONSTS/ INFIN,CEPS,MISVAL,RESCAL,WTPOW
               DO 1 J = 1, NOBS
                  MIN = INFIN
                  LAMBDA = 0.
                  NTMS = 0.
                  DO 2 I = 1, NTRYS
                     TEMP = TRY(J,I,6)
                     A = NOTMIS(TEMP, MISVAL)
                     IF(A.EQ.1)GO TO 3
                     GO TO 4
     3               NTMS = NTMS + 1
                     LAMBDA = LAMBDA + TEMP
                     IF(TEMP.LT.MIN)MIN=TEMP
     4               TEMP = TRY(J,I,4)
                     A = NOTMIS(TEMP,MISVAL)
                     IF(A.EQ.1)GO TO 5
                     GO TO 6
     5               NTMS = NTMS + 1
                     LAMBDA = LAMBDA + TEMP
                     IF(TEMP.LT.MIN)MIN=TEMP
     6               TEMP = TRY(J,I,5)
                     A = NOTMIS(TEMP,MISVAL)
                     IF(A.EQ.1)GO TO 7
                     GO TO 2
     7               NTMS = NTMS + 1
                     LAMBDA = LAMBDA + TEMP
                     IF(TEMP.LT.MIN)MIN = TEMP
     2           CONTINUE
                  LAMBDA = LAMBDA/NTMS
                  LAMBDA = LAMBDA - MIN
                  IF(LAMBDA.GT.0.)LAMBDA = 1./(LAMBDA*RESCAL)
                  IF(MIN.LE.0.)MIN = CEPS
                  DO 8 I = 1, NTRYS
                        TRY(J,I,9) = WEIGHT(TRY(J,I,6), MIN, LAMBDA,J,I)
                        TRY(J,I,7) = WEIGHT(TRY(J,I,4), MIN, LAMBDA,J,I)
                        TRY(J,I,8) = WEIGHT(TRY(J,I,5), MIN, LAMBDA,J,I)
    8             CONTINUE
    1          CONTINUE
         RETURN
         END
C*********************************************************
C
C*********************************************************
         SUBROUTINE COMTRY(NOBS, NTRYS, SMOOTH, TRY)
            DOUBLE PRECISION RSUM,WSUM,T,SMOOTH(NOBS),TRY(NOBS,NTRYS,9)
            REAL CEPS,MISVAL
            INTEGER NOBS, NTRYS, A
            COMMON /CONSTS/ INFIN,CEPS,MISVAL,RESCAL,WTPOW
               DO 1 J = 1, NOBS
                  RSUM = 0.
                  WSUM = 0.
                  DO 2 I = 1, NTRYS
                     T = TRY(J,I,3)
                     A = NOTMIS(T,MISVAL)
                     IF(A.EQ.1)THEN
                        RSUM = RSUM + TRY(J,I,9)*TRY(J,I,3)
                        WSUM = WSUM + TRY(J,I,9)
                     ELSE
                        GO TO 3
                     ENDIF
    3                T = TRY(J,I,1)
                     A = NOTMIS(T,MISVAL)
                     IF(A.EQ.1)THEN
                        RSUM = RSUM + TRY(J,I,7)*TRY(J,I,1)
                        WSUM = WSUM + TRY(J,I,7)
                     ELSE
                        GO TO 4
                     ENDIF
    4                T = TRY(J,I,2)
                     A = NOTMIS(T,MISVAL)
                     IF(A.EQ.1)THEN
                        RSUM = RSUM + TRY(J,I,8)*TRY(J,I,2)
                        WSUM = WSUM + TRY(J,I,8)
                     ELSE
                        GO TO 2
                     ENDIF
    2             CONTINUE
                  IF(WSUM.GE.CEPS)THEN
                     SMOOTH(J) = RSUM/WSUM
                  ELSE
                     SMOOTH(J) = MISVAL
                  ENDIF
    1          CONTINUE
         RETURN
         END
C*******************************************************
C
C*******************************************************
         SUBROUTINE RUNLRC(NOBS, X, Y, W, NTRYS, MNWNSZ, TRYSPN, I, TRY)
            DOUBLE PRECISION AT,BT,MEANRQ,YMEAN,Y2MEAN,
        *   SLOPE,INTER,X(NOBS),Y(NOBS),W(NOBS)
            DOUBLE PRECISION XSUM,YSUM,X2SUM,Y2SUM,XYSUM,
        *   WSUM,XVAR,XMEAN,X2MEAN,XYMEAN,EPS,TRY(NOBS,NTRYS,9)
            REAL CEPS,MISVAL
            INTEGER NOBS, NTRYS, JL, JR, TRYSPN(NTRYS), MNWNSZ,
        *   RFLAG,CFLAG,LFLAG, I, JC
            COMMON /CONSTS/ INFIN,CEPS,MISVAL,RESCAL,WTPOW
               JL = NOBS/4
               JR = 3*JL
               EPS = X(JR) - X(JL)
   2           IF(EPS.LE.0.0.AND.JR.LT.NOBS)THEN
                  IF(JR.LT.NOBS)JR = JR + 1
```

```
             IF(JL.GT.1)JL = JL - 1
             EPS = X(JR) - X(JL)
          ELSE
             GO TO 1
          ENDIF
          GO TO 2
 1        CONTINUE
          EPS = EPS*CEPS
          EPS = EPS**2
          IF(TRYSPN(I).LT.MNWNSZ)TRYSPN(I) = MNWNSZ
          XSUM = 0.
          YSUM = 0.
          WSUM = 0.
          X2SUM = 0.
          Y2SUM = 0.
          XYSUM = 0.
          KI = MNWNSZ - 1
          DO 4 JR = 1, KI
             CALL UPDATE(1, X(JR), Y(JR), W(JR), XSUM, YSUM,
     *       WSUM,X2SUM,Y2SUM, XYSUM)
             TRY(JR,I,1) = MISVAL
 4        TRY(JR,I,4) = MISVAL
          JL = JR - TRYSPN(I) + 1
          JC = JR - TRYSPN(I)/2
          KT = NOBS - MNWNSZ + 1
 6        IF(JL.GE.1)GO TO 7
          LFLAG = 0
          GO TO 8
 7        LFLAG = 1
 8        IF(JC.GE.1.AND.JC.LE.NOBS)GO TO 9
          CFLAG = 0
          GO TO 10
 9        CFLAG = 1
 10       IF(JR.LE.NOBS)GO TO 11
          RFLAG = 0
          GO TO 12
 11       RFLAG = 1
 12       IF(RFLAG.EQ.1)GO TO 13
          GO TO 14
 13       CALL UPDATE(1, X(JR), Y(JR), W(JR), XSUM, YSUM,
     *    WSUM,X2SUM,Y2SUM, XYSUM)
 14       XMEAN = XSUM/WSUM
          X2MEAN = X2SUM/WSUM
          YMEAN = YSUM/WSUM
          XYMEAN = XYSUM/WSUM
          Y2MEAN = Y2SUM/WSUM
          XVAR = X2MEAN - XMEAN**2
          IF(XVAR.LE.EPS)THEN
                SLOPE = 0.
          ELSE
                SLOPE = (XYMEAN - XMEAN*YMEAN)/XVAR
          ENDIF
          INTER = YMEAN - SLOPE*XMEAN
          MEANRQ = Y2MEAN -(2.*INTER*YMEAN)-
     *    (2.*SLOPE*XYMEAN) + (INTER**2) +
     *    (2.*INTER*SLOPE*XMEAN) + (X2MEAN*SLOPE**2)
          IF(LFLAG.EQ.1)GO TO 15
          GO TO 16
 15       AT=TRY(JL,I,3)
          BT=TRY(JL,I,6)
          CALL EVALFT(INTER,SLOPE,MEANRQ,JL,W,X,Y,AT,BT,I,WSUM)
          TRY(JL,I,3)=AT
          TRY(JL,I,6)=BT
 16       IF(RFLAG.EQ.1)GO TO 17
          GO TO 18
 17       AT =TRY(JR,I,1)
          BT =TRY(JR,I,4)
          CALL EVALFT(INTER,SLOPE,MEANRQ,JR,W,X,Y,AT,BT,I,WSUM)
          TRY(JR,I,1)=AT
          TRY(JR,I,4)=BT
```

```
18            IF(CFLAG.EQ.1)GO TO 19
              GO TO 20
19            AT=TRY(JC,I,2)
              BT=TRY(JC,I,5)
              CALL EVALFT(INTER,SLOPE,MEANRQ,JC,W,X,Y,AT,BT,I,WSUM)
              TRY(JC,I,2)=AT
              TRY(JC,I,5)=BT
20            IF(LFLAG.EQ.1)GO TO 21
              GO TO 22
21            CALL UPDATE(0,X(JL),Y(JL),W(JL),XSUM,YSUM,
     *        WSUM,X2SUM,Y2SUM,XYSUM)
22            JR = JR + 1
              JL = JL + 1
              JC = JC + 1
              IF(JL.LE.KT)GO TO 6
              KL = JL
              DO 23 JL = KL, NOBS
              TRY(JL,I,3) = MISVAL
 23           TRY(JL,I,6) = MISVAL
          RETURN
          END
C*******************************************************
C
C*******************************************************
          INTEGER FUNCTION NOTMIS(AO,MISVAL)
            DOUBLE PRECISION AO
            REAL MISVAL
              IF(AO.GT.MISVAL)GO TO 1
              NOTMIS = 0
            RETURN
 1            NOTMIS = 1
            RETURN
            END
C*******************************************************
C
C*************************************************** ****
          REAL FUNCTION WEIGHT(R,INTER2,SLOPE2,J,I)
            DOUBLE PRECISION R,TEMP2,INTER2,SLOPE2
            REAL WTPOW,MISVAL
            INTEGER A,J
            COMMON /CONSTS/ INFIN,CEPS,MISVAL,RESCAL,WTPOW
              A = NOTMIS(R,MISVAL)
              IF(A.EQ.1)THEN
                TEMP2 = SLOPE2*(R - INTER2)
                IF(TEMP2.LE.0.)THEN
                  WEIGHT = 1.
                ELSE
                  IF(TEMP2.LT.1.)THEN
                    WEIGHT = ((1.-TEMP2)**(INT(WTPOW)))*(
     *               (1.-TEMP2)**(WTPOW- INT(WTPOW)))
                  ELSE
                    WEIGHT = 0.
                  ENDIF
                ENDIF
              ELSE
                WEIGHT = 0.
              ENDIF
            RETURN
            END
C********************************************************
C
C********************************************************
          SUBROUTINE UPDATE(OP,A1,B1,C1,XSUM,YSUM,WSUM,X2SUM,
         *Y2SUM,XYSUM)
            DOUBLE PRECISION XSUM,YSUM,WSUM,X2SUM,Y2SUM,XYSUM,A1,B1,C1
            INTEGER OP
              IF(OP.EQ.0)THEN
                XSUM = XSUM - C1*A1
                YSUM = YSUM - C1*B1
                WSUM = WSUM - C1
            X2SUM = X2SUM - C1*A1**2
            Y2SUM = Y2SUM - C1*B1**2
            XYSUM = XYSUM - C1*A1*B1
          ELSE
            XSUM = XSUM + C1*A1
            YSUM = YSUM + C1*B1
            WSUM = WSUM + C1
            X2SUM = X2SUM + C1*A1**2
            Y2SUM = Y2SUM + C1*B1**2
            XYSUM = XYSUM + C1*A1*B1
          ENDIF
        RETURN
        END
C************************************************************
C
C**********************************************************
      SUBROUTINE EVALFT(A,B,R2,JI,W,X,Y,TRYS,TRYR,I,WSUM)
        DOUBLE PRECISION TRYS,TRYR,WSUM,A,B,R2,W(JI),X(JI),
     *Y(JI),FIT,RES
        INTEGER JI,I
          FIT = A + B*X(JI)
          RES = Y(JI) - FIT
          TRYS = FIT
          TRYR = (WSUM*R2 - W(JI)*RES*RES)/(WSUM - W(JI))
      RETURN
      END
C***********************************************************
C
C***********************************************************
      SUBROUTINE SORTER(X,W,Y,N)
      DOUBLE PRECISION X(N),W(N),Y(N),D(5000)
      INTEGER N,KEY(5000)
         DO 5 I=1,N
  5      KEY(I)=I
         CALL SHSORT(X,KEY,N)
         DO 1 I=1,N
  1      D(I) = W(I)
         DO 2 I=1,N
           J=KEY(I)
  2      W(I)=D(J)
         DO 3 I=1,N
  3      D(I)=Y(I)
         DO 4 I=1,N
           J=KEY(I)
  4      Y(I)=D(J)
      RETURN
      END
C**********************************************************
C
C**********************************************************
      BLOCK DATA
      COMMON /CONSTS/ INFIN,CEPS,MISVAL,RESCAL,WTPOW
      REAL INFIN,CEPS,MISVAL,RESCAL,WTPOW
      DATA INFIN,CEPS,MISVAL,RESCAL,WTPOW /1.0E30,1.0E-10,
     *-1.0E30,1.0,2.0/
      END
```

3. *SPTLIN VSAPLWS*

The following two APL functions are used in conjunction with the two files listed above. They were developed by the author of this thesis in order to plot the smoothed data/results. They are the main APL functions within the APL workspace SPTLIN. The first APL function links the user of the smoothing program with GRAFSTAT and gives a user familiar with GRAFSTAT the opportunity to proceed into GRAFSTAT where a greater variety of graphic functions are available.

```
    ∇ ST RT;C
[1]     DUM←CMS 'CLRSCRN'
[2]     'THE ENTIRE DATA FILE THAT YOU WANTED SMOOTHED HAS
BEEN TRANSFERRED'
[3]     'TO THIS WORKSPACE SO THAT YOU MAY BE ABLE TO PLOT
BOTH'
[4]     'THE SMOOTHED AND UNSMOOTHED DATA.'
[5]     'THE UNSMOOTHED DATA IS IN THE VARIABLE WITH THE SAME
NAME AS'
[6]     'THE DATA FILE THAT YOU HAVE YOUR INPUT DATA IN.'
[7]     '            '
[8]     '            '
[9]     'DO YOU WISH TO GO INTO APL OR CONTINUE?'
[10]    'ENTER 0 FOR APL OR 1 FOR CONTINUE'
[11]    C←⍞
[12]    →13+C×8
[13]    'YOU WILL BE SENT TO APL AFTER YOU HAVE READ THIS
IMPORTANT TEXT.'
[14]    '***AFTER YOU HAVE FINISHED WORKING IN APL AND WISH TO
PLOT THE DATA.'
[15]    '***ENTER PLOTER*******NOTICE THAT PLOTER HAS ONLY ONE
T'
[16]    '            '
[17]    '            '
[18]    'NOW ENTER 0 AGAIN'
[19]    C←⍞
[20]    →C
[21]    PLOTER
    ∇
```

The next APL function creates the APL variables to used in the APL 'PLOT' screen. This plotting option is made available to the user through the above APL function. The user can use this APL function to do the plotting or the GRAFSTAT graphics functions. A user need not fully understand how to use the GRAFSTAT plot screen in order to use this function. Several examples are shown with each of the queries so that the user can see what the entry should look like.

```
        ∇
PLOTER;      ;CO;DUM;P;SY;TI;TL;TP;XL;XO;XS;XT;XY;XV;YL;YO;YS;YT;YV;YY
[1]     DUM←CMS 'CLRSCRN'
[2]     'YOU HAVE ACTIVATED THE PLOTTING FUNCTION'
[3]     'IT IS ASSUMED THAT THE USER IS FAMILIAR WITH THE
GRAFSTAT PLOT FUNC.'
[4]     'AND THE AXIS CONTROL FUNCTION'
[5]     'IF YOU RECEIVE (MAKE) AN ERROR MESSAGE DO THE
FOLLOWING'
[6]     '1.  ENSURE THAT VM READ IS DISPLAYED IN LOWER RIGHT
CORNER OF SCREEN'
[7]     '2.  PRESS THE ENTER KEY'
[8]     '3.  ENTER       PAGE'
[9]     'TO UNDERSCORE A LETTER HOLD THE APL/ALT KEY DOWN AND
PRESS THE LETTER'
[10]    'THE PLOTTING FUNCTION WILL RESTART AT THE BEGINNING'
[11]    'THE PLOTTING FUNCTION CAN BE EXITED AT ANY INPUT
POINT BY ENTERING'
[12]    '            →'
```

```
[13]  'AT ANYTIME THAT YOU EXIT THE PLOTTING FUNCTION'
[14]  'YOU WILL BE IN THE GRAFSTAT WORKSPACE'
[15]  'IF YOU WISH TO RETURN TO CMS ENTER'
[16]  '           )OFF HOLD'
[17]  '      '
[18]  '      '
[19]  LB:'ENTER  X  VARIABLE(S)  (ENCLOSED  IN  QUOTES),  IF
ENTERING MORE THAN ONE VARIABLE'
[20]  'SEPARATE VARIABLES WITH SEMICOLON AND USE QUOTES'
[21]  'E.G. ''X'' OR ''X1;X2''  '
[22]  XV←⍞
[23]  DUM←CMS 'CLRSCRN'
[24]  'ENTER Y VARIABLE(S) (ENCLOSED IN QUOTES AND MUST BE
OF SAME LENGTH AS X)
      '
[25]  'IF ENTERING MORE THAN ONE VARIABLE, SEPARATE WITH
SEMICOLON'
[26]  'AND REMEMBER TO USE QUOTES ENCLOSING ENTIRE STRING'
[27]  'E.G. ''Y'' OR ''Y1;Y2''  '
[28]  YV←⍞
[29]  'ENTER  A  VECTOR  INDICATING  TYPE(S)  OF  PLOT;  0≡SYM
ONLY; 1≡LINE ONLY'
[30]  'E.G. 0 OR 1 OR 0 1 OR 0 0 OR 1 0 OR 1 1'
[31]  TP←⍞
[32]  →((×/TP)>0)/L1
[33]  'ENTER  TYPE  OF  SYMBOL  CORRESPONDING  TO  EACH  SYMBOLS
ONLY PLOT (IN QUOTES)
      '
[34]  'E.G. ''.'' OR ''.*'' YOU CAN USE .*+×'
[35]  SY←⍞
[36]  →((+/TP)≡0)/LP
[37] L1:'ENTER A VECTOR INDICATING TYPE(S) OF LINES; 1≡SOLID
LINE; 3≡DASH LINE'
[38]  'E.G. 1 OR 3 OR 1 3 OR ANY OTHER COMBINATION OR LINE
TYPES IN GRAFSTAT'
[39]  TL←⍞
[40] LP:TL←1
[41]  →((ρTP)≥1)/L2
[42]  SY←'.'
[43] L2:DUM←CMS 'CLRSCRN'
[44]  'ENTER SCALE OF X-AXIS (IN QUOTES) OR P (IN QUOTES)
FOR PREVIOUS SCALE'
[45]  'E.G. ''LIN'' OR ''LIN XMIN XMAX'' OR ''P''  '
[46]  XS←⍞
[47]  'ENTER SCALE OF Y-AXIS (IN QUOTES) OR P (IN QUOTES)
FOR PREVIOUS SCALE'
[48]  'E.G. ''LIN'' OR ''LIN YMIN YMAX'' OR ''P''   '
[49]  YS←⍞
[50]  'ENTER THE PLOT HEADER (IN QUOTES) OR EMPTY QUOTES'
[51]  'E.G. ''TITLE'' OR '' ''  '
[52]  TI←⍞
[53]  DUM←CMS 'CLRSCRN'
[54]  'ENTER X-AXIS LABEL (IN QUOTES) OR '
[55]  'A  PAIR  OF  EMPTY  QUOTES  FOR  NO  LABEL  OR  TO  USE  AXIS
CONTROL'
[56]  'E.G. ''LABEL'' OR '' ''  '
[57]  XL←⍞
[58]  'ENTER Y-AXIS LABEL (IN QUOTES) OR'
[59]  'A  PAIR  OF  EMPTY  QUOTES  FOR  NO  LABEL  OR  TO  USE  AXIS
CONTROL'
[60]  'E.G. ''LABLE'' OR '' ''  '
[61]  YL←⍞
[62]  'DO YOU WANT TO RUN THIS PAGE?'
[63]  'ENTER 0 FOR NO OR 1 FOR YES'
[64]  CO←⍞
[65]  DUM←CMS 'CLRSCRN'
[66]  →L3+4×(CO≡0)
[67] L3:P← 1 1 1
[68]     ← 0 1 0 0
[69]  'PLEASE WAIT RUNNING PAGE'
```

```
[70]  RUN PAGESAM
[71]  'DO YOU WANT TO EXIT THIS FUNCTION?'
[72]  'ENTER 0 FOR NO OR 1 FOR YES'
[73]  CO←⎕
[74]  →(CO≡1)/LE
[75]  'DO YOU WANT TO RESTART THIS FUNCTION?'
[76]  'ENTER 0 FOR NO OR 1 FOR YES'
[77]  CO←⎕
[78]  DUM←CMS 'CLRSCRN'
[79]  →(CO≡1)/LB
[80]  'THE ONLY THING LEFT TO DO IS THE AXIS CONTROL'
[81] L6:'WITH THE PARTIAL PLOT THAT YOU HAVE JUST FINISHED
CONSTRUCTING'
[82]  'ENTER  A 3 ELEMENT VECTOR FOR PARTIAL PLOT'
[83]  '1ST ELEMENT, 1(0): LINES AND SYMBOLS ARE (NOT) SHOWN
ON SCREEN'
[84]  '2ND ELEMENT, 1(0): HEADER AND AXES ARE (NOT) SHOWN ON
SCREEN'
[85]  '3RD ELEMENT, 1(0): AXES, GRIDS, AND GRID LINES ARE
(NOT) SHOWN'
[86]  'E.G. 1 1 0 WILL SHOW EVERYTHING ON GRAPH EXCEPT AXES
AND GRID LINES'
[87]  P←⎕
[88]  'ENTER A 4 ELEMENT VECTOR FOR AXES AND GRID CONTROL'
[89]  '1ST ELEMENT, X-AXIS: 0 ≡ BOTTOM, 2 ≡ TOP, OR 20 ≡ AT
Y≡0'
[90]  '2ND ELEMENT, Y-AXIS: 1 ≡ LEFT, 3 ≡ RIGHT, OR 21 ≡ AT
X≡0'
[91]  '3RD  ELEMENT,  VERTICAL  GRID  LINES:  0≡NO  GRID,
1≡DOTTED, OR 2≡SOLID'
[92]  '4TH  ELEMENT,  HORIZON.  GRID  LINES:  0≡NO  GRID,
1≡DOTTED, OR 2≡SOLID'
[93]  'E.G. 2 1 2 2 WILL DISPLAY AXIS AT TOP AND LEFT AND
SOLID GRID LINES'
[94]    ←⎕
[95] L8:'PLEASE WAIT RUNNING PAGE'
[96]  RUN PAGESAM
[97] LA:DUM←CMS 'CLRSCRN'
[98]  'ENTER X-AXIS TIC MARKS LOCATION VECTOR'
[99]  'OR ENTER 0 FOR STANDARD TIC MARKS'
[100]  'OR ENTER 1 FOR NO TIC MARKS'
[101]  'E.G. 1 5 11 OR A VECTOR NAME OR 0 OR 1'
[102]  '1 5 11 WILL SHOW TIC MARKS AT X≡1, X≡5, AND X≡11'
[103]  XT←⎕
[104]  'ENTER X-AXIS SYMBOLS (IN QUOTES)'
[105]  'OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS'
[106]  'OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS'
[107]  'E.G. ''1970;1971'' OR A VECTOR NAME OR 0 OR 1'
[108]  XY←⎕
[109]  'ENTER X-AXIS SYMBOLS LOCATIONS VECTOR'
[110]  'OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO
SYMBOLS'
[111]  'E.G. 6 18 OR A VECTOR NAME OR 0'
[112]  '6 18 WILL SHOW 1970 AT X≡6 AND 1971 AT X≡18'
[113]  XO←⎕
[114]  DUM←CMS 'CLRSCRN'
[115]  'ENTER Y-AXIS TIC MARKS LOCATION VECTOR'
[116]  'OR ENTER 0 FOR STANDARD TIC MARKS'
[117]  'OR ENTER 1 FOR NO TIC MARKS'
[118]  'E.G. ¯1 0 1 OR A VECTOR NAME OR 0 OR 1'
[119]  '¯1 0 1 WILL SHOW TIC MARKS AT Y≡¯1, Y≡0, AND Y≡1'
[120]  YT←⎕
[121]  'ENTER Y-AXIS SYMBOLS (IN QUOTES)'
[122]  'OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS'
[123]  'OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS'
[124]  'E.G. ''LO MID HI'' OR VECTOR NAME OR 0 OR 1'
[125]  YY←⎕
[126]  'ENTER Y-AXIS SYMBOLS LOCATIONS VECTOR'
```

```
[127]  'OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO
SYMBOLS'
[128]  'I.E. ¯1 0 1 OR VECTOR NAME OR 0'
[129]  '¯1 0 1 WILL SHOW LO AT Y≡¯1, MID AT Y≡0, HI AT Y≡1'
[130]  YO←⎕
[131]  'THESE AXIS CONTROL ENTRIES WILL NOW BE RUN'
[132]  RUN PAGEAX
[133]  'DO YOU WANT TO RERUN THE PLOT INPUTS YOU ENTERED'
[134]  'BEFORE RUNNING THIS AXIS CONTROL FUNCTION?'
[135]  'ENTER 0 FOR NO OR 1 FOR YES'
[136]  CO←⎕
[137]  →(CO≡1)/L6
[138]  'DO YOU WANT TO DO ANOTHER AXIS CONTROL PAGE?'
[139]  'ENTER 0 FOR NO OR 1 FOR YES'
[140]  CO←⎕
[141]  →(CO≡1)/L8
[142]  'DO YOU WANT TO RESTART THE FUNCTION?'
[143] LE:'IF YOU DO NOT YOU WILL EXIT THIS FUNCTION'
[144]  'IF YOU EXIT THIS FUNCTION AND WANT TO RETAIN THIS
WORK'
[145]  'USE THE KEEP FUNCTION AND THEN YOU CAN RETURN TO
CMS'
[146]  'BY ENTERING )OFF HOLD'
[147]  'IF YOU WANT TO RETURN TO CMS, SIMPLY ENTER )OFF HOLD
AFTER EXIT'
[148]  'ENTER 0 FOR EXIT OR 1 FOR RESTART'
[149]  CO←⎕
[150]  →(CO≡1)/LB
[151]  ⍳0
  ∇
```

APPENDIX C

APL FUNCTIONS

1. *RANDOM NUMBER GENERATOR*

The following GRAFSTAT [Ref. 12] APL function generates Normal random deviates and was used to produce the N(0,1) noise added to basic functions used in Chapter IV.

```
          ∇ Z←N NORRAND P;S;I;T
[1]       P←2↑P,1
[2]       Z←(N)ρ0
[3]       I←1
[4]     F10:T←2 UNIRAND 0.5 0.5
[5]       T←(2×T)-1
[6]       S←(T[1]*2)+T[2]*2
[7]       →F10×⍳S≥1
[8]       Z[I]←P[1]+P[2]×(T[1]×((¯2×⍟S)÷S)*0.5)
[9]       →F10×⍳(I←I+1)≤N
          ∇
```

The following GRAFSTAT [Ref. 12] APL function generates uniform random numbers which are used in the above APL function, NORRAND.

```
          ∇ R←N UNIRAND B
[1]       R←(B[1]-B[2])+((N?100000000)×2×B[2])÷100000000
          ∇
```

2. *EQUAL-WEIGHT MOVING AVERAGE SMOOTHER*

The following GRAFSTAT [Ref. 12] APL function is the Moving Average smoother used to generate the associated smooth plots in Chapter IV.

```
          ∇ Y←M MOVAV X
[1]       Y←(Mρ÷M) MAV X
          ∇
```

The following GRAFSTAT [Ref. 12] APL function computes weights corresponding to the data values within the neighborhood W and does the weighted averaging within W. These values are the smoothed values which are transferred to the above APL function.

```
          ∇ U←W MAV X;D;J;L
[1]       →3-(1≡ρρW)∧∨/ 1 2 ≡ρρX
[2]       →3+(L≥1)∧(L←ρW)≤1↑D←ρX
[3]       →0,ρ⍞←'NO GO.',U←''
[4]       D[1]←D[1]+1-L
```

```
[5]     U←DρJ←0
[6]     U←U+W[J←J+1]×DρJ∵X
[7]     →6×L>J
        ∇
```

3. COSINE-WEIGHTED MOVING AVERAGE SMOOTHER

The following APL function computes the cosine weights for the cosine-weighted moving average [Ref. 14: p. 394] with window size R, i.e. length:

```
        ∇ W←CW R
[1]     ⍝ WEIGHTS FOR A COSINE-WEIGHTED MOVING AVERAGE OF
LENGTH R
[2]     W←(1-2○(⍳R)×○2÷R+1)÷R+1
        ∇
```

The following APL function is the Cosine Weighted Moving Average algorithm. It is part of the time series APL workspace TSERIES developed at the Naval Postgraduate School.

```
        ∇ S←WW RUNSMOOTH X;L;W;N;M;LW;I;R;IX;IDX
[1]    ⍝ RUNNING SMOOTH OF X WITH WEIGHTS W.
[2]    ⍝ W MUST BE ODD VECTOR THAT ADDS UP TO 1
[3]     ⍝ WW HAS AS 1ST ELEMENT THE ADVANCE STEP L FOR THE
SMOOTHING WINDOW
[4]    ⍝ RESULT IS 2-ROW MATRIX WITH SMOOTH VALUES (ROW 1) AND
INDICES (ROW 2)
[5]     L←⌈/1,1↑WW
[6]     W←1↓WW
[7]     LW←ρW
[8]     IDX←⌊0.5×LW
[9]     →((IDX≠LW÷2)∧(1≡+/W))/L05
[10]    'WEIGHTS W IN RUNSMOOTH NOT ODD OR DONT ADD UP TO 1.
PROGR TERMINATED'
[11]    →
[12]   L05:
[13]    N←ρX
[14]    S←⍳0
[15]    IX←⍳0
[16]    R←¯1+⍳LW
[17]    M←N+1-LW
[18]    I←1-L
[19]   L10:→(M<I←I+L)/L20
[20]    S←S,+/W×X[I+R]
[21]    IX←IX,I+IDX
[22]    →L10
[23]   L20:
[24]   ⍝ RETURN SMOOTHED VALUES AND THEIR INDICES IN ROW 2
[25]    S←(2,ρS)ρS,IX
        ∇
```

APPENDIX D
DATA SETS

This appendix contains in tabular form the data sets used in this thesis. The first table shows the Daily Sea-Surface Temperature in degrees Centigrade collected at Granite Canyon, just south of Point Sur, California. This data set used in Figures 1.1 and 1.2 which were illustrations of a scatterplot and a smooth curve through the scatterplot. The other three tables contain the Test Data sets used in the evaluation of the Supersmoother and the Split Linear Fit smoothers, i.e. Test Set One, Test Set Two, and Test Set Three.

TABLE 19

DAILY SEA-SURFACE TEMPERATURE IN DEGREE CENTIGRADE AT GRANITE CANYON, CALIFORNIA

TEMP.	DATE	TEMP.	DATE	TEMP.	DATE	TEMP.	DATE	TEMP.	DATE	TEMP.	DATE
08.8	1060	08.8	1111	09.7	1162	10.6	1213	13.7	1264	13.2	1315
08.5	1061	08.5	1112	09.9	1163	10.7	1214	14.0	1265	12.4	1316
08.5	1062	09.3	1113	10.1	1164	10.7	1215	14.4	1266	12.5	1317
09.0	1063	09.1	1114	09.8	1165	10.8	1216	14.0	1267	12.0	1318
08.8	1064	09.0	1115	09.8	1166	10.9	1217	14.8	1268	11.4	1319
08.3	1065	09.0	1116	09.5	1167	11.2	1218	14.0	1269	10.8	1320
08.9	1066	09.6	1117	09.8	1168	11.5	1219	13.8	1270	11.3	1321
09.2	1067	09.0	1118	09.8	1169	11.0	1220	13.1	1271	11.1	1322
09.2	1068	09.0	1119	10.2	1170	11.3	1221	12.7	1272	11.3	1323
08.9	1069	09.3	1120	10.5	1171	11.3	1222	12.8	1273	11.9	1324
09.8	1070	09.5	1121	09.9	1172	10.8	1223	11.8	1274	11.9	1325
09.8	1071	10.8	1122	09.9	1173	10.9	1224	11.6	1275	12.1	1326
09.8	1072	10.8	1123	10.0	1174	11.0	1225	11.8	1276	11.7	1327
09.9	1073	10.6	1124	09.8	1175	11.2	1226	11.8	1277	11.7	1328
09.6	1074	10.4	1125	09.7	1176	11.8	1227	12.2	1278	11.5	1329
09.5	1075	09.8	1126	10.2	1177	12.1	1228	11.9	1279	11.0	1330
09.6	1076	09.8	1127	09.9	1178	11.5	1229	11.2	1280	11.5	1331
09.3	1077	10.3	1128	10.0	1179	11.7	1230	11.3	1281	11.5	1332
09.8	1078	10.8	1129	10.0	1180	11.9	1231	11.5	1282	11.9	1333
10.4	1079	10.9	1130	09.9	1181	12.3	1232	11.0	1283	11.2	1334
10.9	1080	10.0	1131	09.9	1182	12.5	1233	10.5	1284	10.9	1335
10.8	1081	10.0	1132	09.8	1183	12.8	1234	10.4	1285	11.2	1336
10.0	1082	10.0	1133	09.8	1184	11.8	1235	11.0	1286	10.8	1337
10.0	1083	09.7	1134	09.9	1185	11.9	1236	11.2	1287	10.9	1338
10.0	1084	10.0	1135	09.5	1186	14.0	1237	11.1	1288	10.7	1339
11.0	1085	09.6	1136	09.9	1187	13.9	1238	11.8	1289	10.0	1340
10.3	1086	08.9	1137	09.9	1188	13.4	1239	12.6	1290	09.6	1341
09.7	1087	08.9	1138	10.4	1189	13.0	1240	13.2	1291	09.2	1342
09.9	1088	08.9	1139	09.9	1190	13.2	1241	14.0	1292	09.6	1343
09.9	1089	09.2	1140	10.5	1191	12.0	1242	13.9	1293	09.7	1344
08.8	1090	08.9	1141	10.2	1192	12.0	1243	12.2	1294	09.8	1345
08.9	1091	09.2	1142	10.0	1193	12.0	1244	12.1	1295	09.4	1346
09.1	1092	09.4	1143	09.2	1194	11.9	1245	11.8	1296	09.8	1347
09.3	1093	10.4	1144	09.5	1195	10.6	1246	11.3	1297	09.8	1348
09.0	1094	10.0	1145	09.8	1196	10.5	1247	11.3	1298	09.7	1349
09.3	1095	09.8	1146	10.0	1197	12.6	1248	11.5	1299	09.8	1350
09.6	1096	10.1	1147	11.2	1198	12.5	1249	10.7	1300	09.5	1351
09.8	1097	10.2	1148	11.2	1199	11.1	1250	09.9	1301	10.0	1352
09.8	1098	10.0	1149	11.7	1200	10.8	1251	09.5	1302	09.9	1353
09.8	1099	09.3	1150	11.8	1201	11.0	1252	10.0	1303	09.9	1354
10.0	1100	09.2	1151	11.2	1202	11.0	1253	10.8	1304	10.0	1355
09.8	1101	08.9	1152	11.2	1203	10.3	1254	09.6	1305	10.6	1356
09.0	1102	09.0	1153	11.2	1204	11.2	1255	10.3	1306	10.7	1357
09.3	1103	09.2	1154	11.2	1205	11.7	1256	10.8	1307	11.0	1358
10.9	1104	09.4	1155	10.9	1206	12.0	1257	11.0	1308	10.8	1359
09.7	1105	09.2	1156	10.9	1207	15.0	1258	11.0	1309	10.5	1360
09.0	1106	09.2	1157	10.9	1208	15.8	1259	11.5	1310	10.8	1361
09.4	1107	09.4	1158	10.6	1209	14.5	1260	12.2	1311	10.7	1362
09.0	1108	09.5	1159	10.4	1210	13.3	1261	11.2	1312	10.7	1363
09.1	1109	09.5	1160	10.3	1211	14.0	1262	12.4	1313	10.7	1364
09.2	1110	09.7	1161	11.4	1212	13.8	1263	11.8	1314	10.7	1365

TABLE 20
TEST SET ONE

X	Y	X	Y	X	Y	X	Y
01	-0.5021245	51	3.5978136	101	3.8639735	151	4.5123774
02	-2.2706658	52	-0.3642989	102	4.6424739	152	7.0449390
03	1.2849535	53	1.0218046	103	3.9925571	153	6.6461625
04	0.7868391	54	3.2826479	104	4.7141715	154	5.9755457
05	0.2750802	55	2.7816672	105	3.1026557	155	6.3987827
06	0.5916070	56	1.9686457	106	3.5086986	156	7.8304268
07	-0.4165126	57	2.6941914	107	5.6847319	157	6.3121917
08	2.0161424	58	1.3421858	108	3.6997858	158	7.2091639
09	0.4190598	59	1.3385338	109	4.5971487	159	5.0608475
10	2.1970718	60	2.7176880	110	2.8131531	160	7.5825731
11	0.7040685	61	3.9561078	111	4.0385152	161	8.2574717
12	1.3516733	62	3.2294325	112	3.7093077	162	5.8956979
13	-0.9261715	63	2.0122998	113	4.2573194	163	5.5093262
14	-0.1411653	64	3.4452995	114	5.5364895	164	5.5995017
15	1.8459821	65	2.3519064	115	5.5778150	165	7.2911596
16	0.0010230	66	1.9137510	116	5.8100211	166	5.8813818
17	1.2573502	67	2.2349597	117	4.8393109	167	6.5830282
18	0.3599704	68	2.1070889	118	5.2195209	168	5.3130510
19	0.6244237	69	2.5508559	119	3.7046249	169	7.7908127
20	-0.5493385	70	3.3621478	120	4.3492568	170	6.0401256
21	-0.4304499	71	1.7760771	121	6.1103783	171	7.7141272
22	1.8645709	72	3.2315889	122	5.7786937	172	7.6411270
23	0.8751194	73	4.0529999	123	5.3587051	173	6.7540765
24	0.1610555	74	3.1099944	124	3.7126557	174	7.2609074
25	0.2348275	75	3.7031440	125	5.3246669	175	6.6775327
26	1.9017348	76	2.9875884	126	5.4300705	176	6.6715890
27	1.0237749	77	5.0984962	127	4.6748617	177	8.2278953
28	1.6334782	78	4.0441594	128	5.4123149	178	7.1614303
29	1.5566808	79	1.3458853	129	7.7259102	179	6.0619503
30	1.9562190	80	3.2349733	130	4.7421844	180	8.7667237
31	1.6404860	81	1.4321380	131	3.6291729	181	6.1915765
32	-0.0613807	82	4.3081925	132	2.6104761	182	7.7027237
33	1.6950412	83	3.7146003	133	4.5603033	183	6.4755850
34	2.4851618	84	3.9994056	134	4.6852791	184	7.0483700
35	2.1286416	85	4.2742129	135	4.2283128	185	7.7978105
36	-0.9374542	86	5.1924022	136	7.4729300	186	8.4897860
37	1.2062176	87	3.1599492	137	6.4484810	187	7.1392044
38	1.1970600	88	3.3825862	138	6.1902920	188	7.8562970
39	1.8779879	89	4.1757696	139	5.9801460	189	7.3386392
40	1.0888278	90	4.5778941	140	2.7272488	190	7.6166495
41	1.6379587	91	2.5246523	141	7.3258741	191	6.4476388
42	3.2865109	92	3.1299786	142	5.7079245	192	7.5483537
43	2.5676486	93	3.7598849	143	4.8179694	193	9.2075244
44	2.0281008	94	1.2771573	144	3.7067425	194	6.9231787
45	0.8765109	95	4.9586547	145	5.8890863	195	6.4990181
46	1.7695006	96	2.8137205	146	7.9270991	196	9.4141318
47	2.0278913	97	5.0334870	147	6.2451185	197	9.0460399
48	1.3629063	98	3.1335435	148	6.7661055	198	8.7064297
49	1.6232943	99	4.5948086	149	5.7754624	199	6.4983612
50	2.4152275	100	4.8204097	150	7.0307144	200	9.5544659

TABLE 21
TEST SET TWO

X	Y	X	Y	X	Y	X	Y
01	0.4570756	51	1.1056374	101	-0.7988832	151	-0.5570467
02	-1.3538641	52	-2.9317810	102	-0.0285995	152	1.9443678
03	2.1577621	53	-1.6202036	103	-0.6857874	153	1.5128773
04	1.6140663	54	0.5669487	104	0.0294491	154	0.8079680
05	1.0551468	55	-0.0068339	105	-1.5876051	155	1.1953248
06	1.3229449	56	-0.8917159	106	-1.1863163	156	2.5894945
07	0.2645429	57	-0.2370383	107	0.9856908	157	1.0321866
08	2.6453778	58	-1.6588701	108	-1.0026112	158	1.8884863
09	0.9949566	59	-1.7312588	109	-0.1079922	159	-0.3021013
10	2.7181328	60	-0.4197057	110	-1.8941798	160	2.1757581
11	1.1688202	61	0.7522926	111	-0.6705179	161	2.8052014
12	1.7586682	62	-0.0395823	112	-1.0009952	162	0.3963923
13	-0.5783523	63	-1.3206523	113	-0.4538849	163	-0.0385836
14	0.1460898	64	0.0497107	114	0.8246897	164	0.0014325
15	2.0713177	65	-1.1049823	115	0.8656625	165	1.6413922
16	0.1631187	66	-1.6030669	116	1.0976953	166	0.1783960
17	1.3549229	67	-1.3403846	117	0.1269276	167	0.8253250
18	0.3917761	68	-1.5253494	118	0.5071318	168	-0.5008452
19	0.5892597	69	-1.1372168	119	-1.0077820	169	1.9192740
20	-0.6526318	70	-0.3800745	120	-0.3632442	170	0.1095231
21	-0.6029871	71	-2.0187875	121	1.3976432	171	1.7230700
22	1.6217221	72	-0.6143904	122	1.0655210	172	1.5882570
23	0.5609396	73	0.1574513	123	0.6448279	173	0.6380705
24	-0.2254246	74	-0.8335629	124	-1.0022558	174	1.0804791
25	-0.2248702	75	-0.2868485	125	0.6083290	175	0.4314349
26	1.3679551	76	-1.0472555	126	0.7118525	176	0.3586158
27	0.4151033	77	1.0203924	127	-0.0457512	177	1.8468834
28	0.9491606	78	-0.0756075	128	0.6887321	178	0.7112616
29	0.7960204	79	-2.8139453	129	2.9987233	179	-0.4584467
30	1.1185767	80	-0.9633214	130	0.0107011	180	2.1750750
31	0.7252823	81	-2.8030239	131	-1.1073564	181	-0.4722970
32	-1.0546655	82	0.0377556	132	-2.1319047	182	0.9657036
33	0.6232166	83	-0.5895273	133	-0.1887889	183	-0.3354497
34	1.3344005	84	-0.3368381	134	-0.0714380	184	0.1625056
35	0.8986088	85	-0.0925853	135	-0.5369943	185	0.8363578
36	-2.2470305	86	0.7965962	136	2.6980175	186	1.4520438
37	-0.1831108	87	-1.2633354	137	1.6628992	187	0.0245292
38	-0.2721655	88	-1.0666680	138	1.3929299	188	0.6641043
39	0.3287841	89	-0.2979676	139	1.1698476	189	0.0684045
40	-0.5403718	90	0.0811356	140	-2.0971853	190	0.2679094
41	-0.0711898	91	-1.9936927	141	2.4860636	191	-0.9800088
42	1.4975242	92	-1.4085479	142	0.8514574	192	0.0414590
43	0.6989979	93	-0.7974495	143	-0.0564717	193	1.6211055
44	0.0800240	94	-3.2976455	144	-1.1870254	194	-0.7429779
45	-1.1506912	95	0.3676870	145	0.9746058	195	-1.2470265
46	-0.3364633	96	-1.7921467	146	2.9904894	196	1.5881130
47	-0.1564088	97	0.4139455	147	1.2849344	197	1.1400245
48	-0.8992433	98	-1.4984893	148	1.7808755	198	0.7204593
49	-0.7161574	99	-0.0485762	149	0.763691	199	-1.5674586
50	-0.0009194	100	0.1667661	150	1.9908847	200	1.4089659

TABLE 22

TEST SET THREE

X	Y	X	Y	X	Y	X	Y
01	0.5178755	51	5.4778136	101	-0.1760265	151	-1.1356226
02	-1.2306658	52	1.3957011	102	0.5704739	152	1.3569390
03	2.3449535	53	2.6618046	103	-0.1114429	153	0.9181625
04	1.8668391	54	4.8026479	104	0.5781715	154	0.2075457
05	1.3750802	55	4.1816672	105	-1.0653443	155	0.5907827
06	1.7116070	56	3.2486457	106	-0.6913014	156	1.9824268
07	0.7234874	57	3.8541914	107	1.4527319	157	0.4241917
08	3.1761424	58	2.3821858	108	-0.5642142	158	1.2811639
09	1.5990598	59	2.2585338	109	0.3011487	159	-0.9071525
10	3.3970718	60	3.5176880	110	-1.5148469	160	1.5745731
11	1.9240685	61	4.6361078	111	-0.3214848	161	2.2094717
12	2.5916733	62	3.7894325	112	-0.6826923	162	-0.1923021
13	0.3338285	63	2.4522998	113	-0.1666806	163	-0.6186738
14	1.1388347	64	3.7652995	114	1.0804895	164	-0.5684983
15	3.1459821	65	2.5519064	115	1.0898150	165	1.0831596
16	1.3210230	66	1.9937510	116	1.2900211	166	-0.3666182
17	2.5973502	67	2.1949597	117	0.2873109	167	0.2950282
18	1.7199704	68	1.9470889	118	0.6355209	168	-1.0149490
19	2.0044237	69	2.2708559	119	-0.9113751	169	1.4228127
20	0.8506615	70	2.9621478	120	-0.2987432	170	-0.3678744
21	0.9895501	71	1.2560771	121	1.4303783	171	1.2661272
22	3.3045709	72	2.5915889	122	1.0666937	172	1.1531270
23	2.3351194	73	3.2929999	123	0.6147051	173	0.2260765
24	1.6410555	74	2.2299944	124	-1.0633443	174	0.6929074
25	1.7348275	75	2.7031440	125	0.5166669	175	0.0695327
26	3.4217348	76	1.8675884	126	0.5900705	176	0.0235890
27	2.5637749	77	3.8584962	127	-0.1971383	177	1.5398953
28	3.1934782	78	2.6841594	128	0.5083149	178	0.4334303
29	3.1366808	79	-0.1341147	129	2.7899102	179	-0.7060497
30	3.5562190	80	1.6349733	130	-0.2258156	180	1.9587237
31	3.2604860	81	-0.2878620	131	-1.3708271	181	-0.6564235
32	1.5786193	82	2.4681925	132	-2.4215239	182	0.8147237
33	3.3550412	83	1.7546003	133	-0.5036967	183	-0.4524144
34	4.1651618	84	1.9194056	134	-0.4107209	184	0.0803703
35	3.8286416	85	2.0742129	135	-0.8996872	185	0.7898105
36	0.7825458	86	2.8724022	136	2.3129300	186	1.4417860
37	2.9462176	87	0.7199492	137	1.2564810	187	0.0512044
38	2.9570600	88	0.8225862	138	0.9662920	188	0.7282970
39	3.6579879	89	1.4957696	139	0.7241460	189	0.1706392
40	2.8888278	90	1.7778941	140	-2.5607512	190	0.4086495
41	3.4579587	91	-0.3953477	141	2.0058741	191	-0.8003612
42	5.1265109	92	0.0899786	142	0.3559245	192	0.2603537
43	4.4276486	93	0.5998849	143	-0.5660306	193	1.8795244
44	3.9081008	94	-2.0028427	144	-1.7092575	194	-0.4448213
45	2.7765109	95	1.5586547	145	0.4410863	195	-0.9089819
46	3.6895006	96	-0.7062795	146	2.4470991	196	1.9661318
47	3.9678913	97	1.3934870	147	0.7331185	197	1.5580399
48	3.3229063	98	-0.6264565	148	1.2221055	198	1.1784297
49	3.6032943	99	0.7148086	149	0.1994624	199	-1.0696388
50	4.4152275	100	0.8204097	150	1.4227144	200	1.9464659

APPENDIX E

SAMPLE SESSION USING SUPSMO PROGRAM

The following is a computer session showing the interaction between the smoothing program SUPSMO and a user. The steps necessary to run the smoothing program SPTLIN are very similar to those in this session. It should be remembered that SPTLIN requires a terminal which can access 2M of computer storage memory.

```
 supsmo

 YOU HAVE INITIATED AN ALGORITHM TO SMOOTH A SET OF DATA
USING THE
 ALGORITHM  "SUPER  SMOOTHER"  DEVELOPED  BY  FRIEDMAN  AND
STUETZLE OF
 STANFORD UNIVERSITY DEPT. OF STATISTICS

 IF GRAPHICS WILL NOT BE USED DEFINE STORAGE AS 1024K
 BY ENTERING 'DEF STOR 1024K'
 FOLLOWED BY 'I CMS',
 THEN BY 'SUPSMO'

 DO YOU WISH TO CONTINUE?
 ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:


 Y


 IN ORDER TO USE THIS ALGORITHM YOU MUST HAVE ON HAND THE
FOLLOWING:

 1. FILENAME OF DATA FILE (FILETYPE DATA) WITH DATA TO BE
SMOOTHED

 2. IF DATA POINTS ARE NOT IN CHRONOLOGICAL ORDER, YOU NEED
TO
 HAVE A FILE (FILETYPE ORDER) WITH INDICES CORRESPONDING TO
DATA
 POINTS INDICATING THE ORDER OF THE DATA POINTS.

 3. FILENAME OF DATA FILE WHERE SMOOTHED OUTPUT WILL BE
WRITTEN
 OR IF YOU WANT TO WRITE OUTPUT INTO APL HAVE ON HAND
 THE VARIABLE AND WORKSPACE NAMES THAT WILL STORE THE
OUTPUT.

 4. IF YOU WANT TO SMOOTH THE DATA USING ONLY ONE WINDOW
SIZE,
 HAVE ON HAND THE DECIMAL FRACTION OF THE DATA TO BE USED.

 5. IF YOU WANT TO SMOOTH THE DATA USING THREE WINDOW SIZES,
 HAVE ON HAND THE THREE DECIMAL FRACTIONS OF THE DATA TO BE
USED.

 DO YOU WISH TO CONTINUE?
 ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:
```

```
 Y


 ENTER  FILENAME  OF  FILE  WHICH  CONTAINS  THE  DATA  TO  BE
SMOOTHED:


 water


 ARE DATA POINTS TO BE SMOOTHED IN CHRONOLOGICAL ORDER?
 ENTER Y FOR YES OR N FOR NO:


 Y


 THE DATA YOU WANT TO SMOOTH IS IN WATER DATA
 WHERE DO YOU WANT TO WRITE THE SMOOTHED OUTPUT? CMS OR APL?
 YOU CAN PLOT THE SMOOTHED OUTPUT IF YOU ARE LOGGED ON
 A  TERMINAL  THAT  CAN  ACCESS  GRAFSTAT,  I.E.  HAVE  2M  OF
STORAGE
 BUT THE OUTPUT MUST BE STORED IN AN APL VARIABLE
 ENTER APL OR CMS:


 apl


 NOT USING THE NAME OF THE FILE WITH THE INPUT DATA, WATER
 ENTER  THE  NAME  OF  THE  APL  VARIABLE  THAT  WILL  STORE  THE
OUTPUT:


 smufig53

 DO YOU WANT TO PLOT THE OUTPUT?
 ENTER Y FOR YES OR N FOR NO:


 Y


 CAN YOU ACCESS 2M OF STORAGE ON THIS DISK (TERMINAL)?
 ENTER Y FOR YES OR N FOR NO:


 Y


 PLEASE READ THE FOLLOWING INSTRUCTIONS VERY CAREFULLY
 ARE YOU READY TO START THE SUPER SMOOTHING PROGRAM?
 ENTER Y FOR YES OR ANY OTHER KEY TO EXIT:


 Y


 *****PLEASE   WAIT   THE   SMOOTHING   PROGRAM   IS   BEING
COMPILED**********

 VS FORTRAN COMPILER ENTERED.  16:11:30

 **MAIN** END OF COMPILATION 1 ******

 **SUPSMU** END OF COMPILATION 2 ******
 **SMOOTH** END OF COMPILATION 3 ******

 **SORTER** END OF COMPILATION 4 ******

 **BLKDT#** END OF COMPILATION 5 ******
 VS FORTRAN COMPILER EXITED.   16:11:36


 *****PLEASE    WAIT    SMOOTHING    PROGRAM    IS    BEING
LOADED*****************

 EXECUTION BEGINS . . .
  ENTER THE NUMBER OF DATA POINTS TO BE SMOOTHED---INTEGER
VALUE


 671


  ARE THE INPUT DATA POINTS IN CHRONOLOGICAL ORDER?
  ENTER 0 FOR NO OR 1 FOR YES


 1


  ENTER 1.0 IF YOU DESIRE TO USE ONLY ONE SPAN VALUE
 ENTER 0.0 IF YOU DESIRE TO USE THREE SPAN VALUES


 0.0


  ENTER THE LOWEST SPAN VALUE:
  FRACTION OF   671 I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0


 0.00745


  ENTER THE MIDDLE SPAN VALUE:
  FRACTION OF   671 I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0


 0.016393


  ENTER THE HIGHEST SPAN VALUE:
  FRACTION OF   671 I.E. A REAL NUMBER BETWEEN 0.0 AND 1.0


 0.0175


  IF ONE OF THE SPAN VALUES IS SMALL
 I.E. RESULTS    IN A SMALL WINDOW SIZE (10 OR LESS)
 YOU MAY WISH TO ADJUST THE ROBUSTNESS
 BY ENTERING A REAL NUMBER GT 0.0 BUT LT 10.0
 OR FOR NO ROBUST ADJUSTMENT ENTER 0.0
 ENTER YOUR CHOICE


 0.0



 *****PLEASE WAIT SMOOTHING PROGRAM NOW RUNNING******

 ****PLEASE WAIT, LINKING TO GRAFSTAT****************
 N (193) R/O
 M (194) R/O
 F (391) R/O

       V S  A P L   4.0

 CLEAR WS
 SAVED 06:26:45 09/01/85
 WSSIZE IS 1188956

 CMS MATRIX IS NOT RECTANGULAR.  ROW ONE HAS 5 ELEMENTS
 ROW 135 HAS 1 ELEMENT(S)
 INFORMATION TRANSFER HAS STOPPED AT THIS LINE

 16:12:42 09/04/85 SUPSMO
 SAVED 15:18:53 05/09/85
 WSSIZE IS 1188956

 THIS IS THE 4/01/85 RELEASE OF GRAFSTAT.  IT RUNS ON THE
3277/GA OR
 ON THE 3278/79.  CONTROL VECTORS FROM EARLIER RELEASES WILL
CONTINUE
 TO RUN.  IF YOU )COPY RATHER THAN )LOAD THIS WORKSPACE YOU
MUST
 EXECUTE  THE  FUNCTIONL  TENT   BEFORE  STARTING.   THE  NEXT
RELEASE IS
 SCHEDULED FOR 9/85.

 TO BEGIN, TYPE:  START

 FOR MORE INFORMATION, TYPE:  DESCRIBE

 NOT COPIED: RCODE  GET XBLANKS VCAT
 SAVED 16:12:42 09/04/85
 THEENTIRE   DATA  FILE  THAT  YOU  WANTED  SMOOTHED  HAS  BEEN
TRANSFERRED
 TO THIS WORKSPACE SO THAT YOU MAY BE ABLE TO PLOT BOTH
 THE SMOOTHED AND UNSMOOTHED DATA.
 THE UNSMOOTHED DATA IS IN THE VARIABLE WITH THE SAME NAME
AS
 THE DATA FILE THAT YOU HAVE YOUR INPUT DATA IN.


 DO YOU WISH TO GO INTO APL OR CONTINUE?
 ENTER 0 FOR APL OR 1 FOR CONTINUE ⍞ :


       1


 YOU HAVE ACTIVATED THE PLOTTING FUNCTION
 IT IS ASSUMED THAT THE USER IS FAMILIAR WITH THE GRAFSTAT
PLOT FUNC.
 AND THE AXIS CONTROL FUNCTION
 IF YOU RECEIVE (MAKE) AN ERROR MESSAGE DO THE FOLLOWING
 1.  ENSURE THAT VM READ IS DISPLAYED IN LOWER RIGHT CORNER
OF SCREEN
 2.  PRESS THE ENTER KEY
 3.  ENTER      P̲ AG
 TO UNDERSCORE A LETTER HOLD THE APL/ALT KEY DOWN AND PRESS
THE LETTER
 THE PLOTTING FUNCTION WILL RESTART AT THE BEGINNING
 THE PLOTTING FUNCTION CAN BE EXITED AT ANY INPUT POINT BY
ENTERING
              →
 AT ANYTIME THAT YOU EXIT THE PLOTTING FUNCTION
 YOU WILL BE IN THE GRAFSTAT WORKSPACE
 IF YOU WISH TO RETURN TO CMS ENTER
             )OFF HOLD
 ENTER X VARIABLE(S) (ENCLOSED IN QUOTES), IF ENTERING MORE
THAN ONE VARIABLE
 SEPARATE VARIABLES WITH SEMICOLON AND USE QUOTES
 E.G. 'X' OR 'X1;X2' ⍞ :

       '( 670)+59'

 ENTER Y VARIABLE(S) (ENCLOSED IN QUOTES AND MUST BE OF SAME
LENGTH AS X)
 IF ENTERING MORE THAN ONE VARIABLE, SEPARATE WITH SEMICOLON
 AND REMEMBER TO USE QUOTES ENCLOSING ENTIRE STRING
 E.G. 'Y' OR 'Y1;Y2' ⍞ :

       '670 WATER;SMUFIG53'

 ENTER A VECTOR INDICATING TYPE(S) OF PLOT; 0≡SYM ONLY;
1≡LINE ONLY
 E.G. 0 OR 1 OR 0 1 OR 0 0 OR 1 0 OR 1 1 ⍞ :

       0 1

 ENTER TYPE OF SYMBOL CORRESPONDING TO EACH SYMBOLS ONLY
PLOT (IN QUOTES)
 E.G. '.' OR '.*' YOU CAN USE .*+ ⍞ :

       '.'

 ENTER A VECTOR INDICATING TYPE(S) OF LINES; 1≡SOLID LINE;
3≡DASH LINE
 E.G. 1 OR 3 OR 1 3 OR ANY OTHER COMBINATION OR LINE TYPES
IN GRAFSTAT ⍞ :

       1

 ENTER SCALE OF X-AXIS (IN QUOTES) OR P (IN QUOTES) FOR
PREVIOUS SCALE
 E.G. 'LIN' OR 'LIN XMIN XMAX' OR 'P' ⍞ :

       'LIN¯ .1 735'

 ENTER SCALE OF Y-AXIS (IN QUOTES) OR P (IN QUOTES) FOR
PREVIOUS SCALE
 E.G. 'LIN' OR 'LIN YMIN YMAX' OR 'P' ⍞ :

       'LIN 8 17'

 ENTER THE PLOT HEADER (IN QUOTES) OR EMPTY QUOTES
 E.G. 'TITLE' OR ' ' ⍞ :

       'SMOOTHING WITH SUPERSMOOTHER, ALPHA≡ 0.0¨ SPAN(S)≡
0.00745, 0.016393, 0.0175'
```

```
 ENTER X-AXIS LABEL (IN QUOTES) OR
 A PAIR OF EMPTY QUOTES FOR NO LABEL OR TO USE AXIS CONTROL
 E.G. 'LABEL' OR ' ' ⍞ :

       'JULIAN CALENDAR DATE'

 ENTER Y-AXIS LABEL (IN QUOTES) OR
 A PAIR OF EMPTY QUOTES FOR NO LABEL OR TO USE AXIS CONTROL
 E.G. 'LABLE' OR ' ' ⍞ :

       'TEMPERATURE IN DEGREES CENT.'

 DO YOU WANT TO RUN THIS PAGE?
 ENTER 0 FOR NO OR 1 FOR YES ⍞ :

       0

 DO YOU WANT TO EXIT THIS FUNCTION?
 ENTER 0 FOR NO OR 1 FOR YES

⍞ :
       0

 DO YOU WANT TO RESTART THIS FUNCTION?
 ENTER 0 FOR NO OR 1 FOR YES ⍞ :

       0

 THE ONLY THING LEFT TO DO IS THE AXIS CONTROL
 WITH  THE  PARTIAL  PLOT  THAT  YOU  HAVE  JUST  FINISHED
CONSTRUCTING
 ENTER  A 3 ELEMENT VECTOR FOR PARTIAL PLOT
 1ST  ELEMENT, 1(0): LINES AND SYMBOLS ARE (NOT) SHOWN ON
SCREEN
 2ND  ELEMENT,  1(0):  HEADER  AND  AXES  ARE  (NOT)  SHOWN  ON
SCREEN
 3RD ELEMENT, 1(0): AXES, GRIDS, AND GRID LINES ARE (NOT)
SHOWN
 E.G. 1 1 0 WILL SHOW EVERYTHING ON GRAPH EXCEPT AXES AND
GRID LINES ⍞ :

       1 1 0

 ENTER A 4 ELEMENT VECTOR FOR AXES AND GRID CONTROL
 1ST ELEMENT, X-AXIS: 0 ≡ BOTTOM, 2 ≡ TOP, OR 20 ≡ AT Y≡0
 2ND ELEMENT, Y-AXIS: 1 ≡ LEFT, 3 ≡ RIGHT, OR 21 ≡ AT X≡0
 3RD ELEMENT, VERTICAL GRID LINES: 0≡NO GRID, 1≡DOTTED, OR
2≡SOLID
 4TH ELEMENT, HORIZON. GRID LINES: 0≡NO GRID, 1≡DOTTED, OR
2≡SOLID
 E.G. 2 1 2 2 WILL DISPLAY AXIS AT TOP AND LEFT AND SOLID
GRID LINES ⍞ :

       0 1 0 0

 PLEASE WAIT RUNNING PAGE
 ENTER X-AXIS TIC MARKS LOCATION VECTOR
 OR ENTER 0 FOR STANDARD TIC MARKS
 OR ENTER 1 FOR NO TIC MARKS
 E.G. 1 5 11 OR A VECTOR NAME OR 0 OR 1
 1 5 11 WILL SHOW TIC MARKS AT X≡1, X≡5, AND X≡11 ⍞ :


       0 31 59 90 120 151 181 212 243 273 304 334 365 396
424 455 485 516 546 577 608 638 669 699 730


 ENTER X-AXIS SYMBOLS (IN QUOTES)
 OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS
 OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS
 E.G. '1970;1971' OR A VECTOR NAME OR 0 OR 1 ⍞ :

       '1080;1172;1264;1355;2080;2172;2264;2355'


 ENTER X-AXIS SYMBOLS LOCATIONS VECTOR
 OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO SYMBOLS
 E.G. 6 18 OR A VECTOR NAME OR 0
 6 18 WILL SHOW 1970 AT X≡6 AND 1971 AT X≡18 ⍞ :

       80 172 264 355 445 536 629 721


 ENTER Y-AXIS TIC MARKS LOCATION VECTOR
 OR ENTER 0 FOR STANDARD TIC MARKS
 OR ENTER 1 FOR NO TIC MARKS
 E.G.¯ 1 0 1 OR A VECTOR NAME OR 0 OR 1 ¯ 1 0 1 WILL SHOW
TIC MARKS AT Y≡ 1, Y≡0, AND Y≡1 ⍞ :

       0


 ENTER Y-AXIS SYMBOLS (IN QUOTES)
 OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS
 OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS
 E.G. 'LO MID HI' OR VECTOR NAME OR 0 OR 1 ⍞ :

       0


 ENTER Y-AXIS SYMBOLS LOCATIONS VECTOR
 OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO SYMBOLS
 I.E.¯ 1 0 1 OR VECTOR NAME OR 0 ¯ 1 0 1 WILL SHOW LO AT Y≡
1, MID AT Y≡0, HI AT Y≡1 ⍞ :

       0


 THESE AXIS CONTROL ENTRIES WILL NOW BE RUN


 DO YOU WANT TO RERUN THE PLOT INPUTS YOU ENTERED
 BEFORE RUNNING THIS AXIS CONTROL FUNCTION?
 ENTER 0 FOR NO OR 1 FOR YES ⍞ :

       0
 DO YOU WANT TO RERUN THE PLOT INPUTS YOU ENTERED
 BEFORE RUNNING THIS AXIS CONTROL FUNCTION?
 ENTER 0 FOR NO OR 1 FOR YES ⍞ :

       1

 WITH   THE   PARTIAL   PLOT   THAT   YOU   HAVE   JUST   FINISHED
CONSTRUCTING
 ENTER  A 3 ELEMENT VECTOR FOR PARTIAL PLOT
 1ST  ELEMENT,  1(0):  LINES  AND  SYMBOLS  ARE  (NOT)  SHOWN  ON
SCREEN
 2ND  ELEMENT,  1(0):  HEADER  AND  AXES  ARE  (NOT)  SHOWN  ON
SCREEN
 3RD ELEMENT,  1(0):  AXES,  GRIDS,  AND  GRID  LINES  ARE  (NOT)
SHOWN
 E.G.  1  1  0 WILL  SHOW  EVERYTHING  ON  GRAPH  EXCEPT  AXES  AND
GRID LINES ⍞ :

       0 0 0

 ENTER A 4 ELEMENT VECTOR FOR AXES AND GRID CONTROL
 1ST ELEMENT, X-AXIS: 0 ≡ BOTTOM, 2 ≡ TOP, OR 20 ≡ AT Y≡0
 2ND ELEMENT, Y-AXIS: 1 ≡ LEFT, 3 ≡ RIGHT, OR 21 ≡ AT X≡0
 3RD  ELEMENT,  VERTICAL  GRID  LINES:  0≡NO  GRID,  1≡DOTTED,  OR
2≡SOLID
 4TH  ELEMENT,  HORIZON.  GRID  LINES:  0≡NO  GRID,  1≡DOTTED,  OR
2≡SOLID
 E.G.  2  1  2  2  WILL  DISPLAY  AXIS  AT  TOP  AND  LEFT  AND  SOLID
GRID LINES ⍞ :

       20 3 1 1

 PLEASE WAIT RUNNING PAGE

 ENTER X-AXIS TIC MARKS LOCATION VECTOR
 OR ENTER 0 FOR STANDARD TIC MARKS
 OR ENTER 1 FOR NO TIC MARKS
 E.G. 1 5 11 OR A VECTOR NAME OR 0 OR 1
 1 5 11 WILL SHOW TIC MARKS AT X≡1, X≡5, AND X≡11 ⍞ :

       80 170 260 350 440 530 620 710

 ENTER X-AXIS SYMBOLS (IN QUOTES)
 OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS
 OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS
 E.G. '1970;1971' OR A VECTOR NAME OR 0 OR 1 ⍞ :

       1

 ENTER X-AXIS SYMBOLS LOCATIONS VECTOR
 OR ENTER 0 FOR SYMBOLS AT DEFAULT LOCATIONS OR NO SYMBOLS
 E.G. 6 18 OR A VECTOR NAME OR 0
 6 18 WILL SHOW 1970 AT X≡6 AND 1971 AT X≡18 ⍞ :

       0
 ENTER Y-AXIS TIC MARKS LOCATION VECTOR
 OR ENTER 0 FOR STANDARD TIC MARKS
 OR ENTER 1 FOR NO TIC MARKS
 E.G. ¯1 0 1 OR A VECTOR NAME OR 0 OR 1 ¯1 0 1 WILL SHOW
TIC MARKS AT Y≡ 1, Y≡0, AND Y≡1 ⍂ :

       1

 ENTER Y-AXIS SYMBOLS (IN QUOTES)
 OR ENTER 0 WITHOUT QUOTES FOR STANDARD SYMBOLS
 OR ENTER 1 WITHOUT QUOTES FOR NO SYMBOLS
 E.G. 'LO MID HI' OR VECTOR NAME OR 0 OR 1 ⍂ :

       1

 ENTER Y-AXIS SYMBOLS LOCATIONS VECTOR
 OR ENTER 0 FOR SYMBOLS AT DEFAULT_LOCATIONS OR NO SYMBOLS
 I.E. ¯1 0 1 OR VECTOR NAME OR 0 ¯1 0 1 WILL SHOW LO AT Y≡
1, MID AT Y≡0, HI AT Y≡1 ⍂ :

       0

 THESE AXIS CONTROL ENTRIES WILL NOW BE RUN
 DO YOU WANT TO RERUN THE PLOT INPUTS YOU ENTERED
 BEFORE RUNNING THIS AXIS CONTROL FUNCTION?
 ENTER 0 FOR NO OR 1 FOR YES ⍂ :

       0

 DO YOU WANT TO DO ANOTHER AXIS CONTROL PAGE?
 ENTER 0 FOR NO OR 1 FOR YES ⍂ :

       0

 DO YOU WANT TO RESTART THE FUNCTION?
 IF YOU DO NOT YOU WILL EXIT THIS FUNCTION
 IF YOU EXIT THIS FUNCTION AND WANT TO RETAIN THIS WORK
 USE THE KEEP FUNCTION AND THEN YOU CAN RETURN TO CMS
 BY ENTERING )OFF HOLD
 IF YOU WANT TO RETURN TO CMS, SIMPLY ENTER )OFF HOLD AFTER
EXIT
 ENTER 0 FOR EXIT OR 1 FOR RESTART ⍂ :

       0
```

LIST OF REFERENCES

1. Bevington, P. R., *Data Reduction and Error Analysis For the Physical Sciences*, McGraw-Hill Book Co., 1969.

2. Tukey, J. W., *Exploratory Data Analysis*, Addison-Wesley Pub. Co., 1977.

3. Chambers, J. M., and others, *Graphical Methods For Data Analysis*, Duxbury Press, 1983.

4. Naval Postgraduate School Report Number NPS55-84-012, *Interannual Variability in Sea-Surface Temperature at One Location Along the Central California Coast*, by L. C. Breaker, P. A. W. Lewis, and E. J. Orav, May 1984.

5. Moore, P. G., *Principles of Statistical Techniques*, Cambridge University Press, 1969.

6. Sadan, S. and Ronen, J., *Smoothing Income Numbers: Objectives, Means, and Implications*, Addison-Wesley, 1981.

7. Blackman, R. B., *Linear Data-Smoothing and Prediction In Theory and Practice*, Addison-Wesley, 1965.

8. *Time Series Analysis: Proceedings of the International Conference* Held at Houston, TX, August 1980, edited by O. D. Anderson and M. R. Perryman, North Holland Pub. Co., Amsterdam, 1981.

9. Stanford Linear Accelerator Center Pub-3013, *Smoothing of Scatterplots*, by J. H. Friedman and W. Stuetzle, July 1982, revised July 1984.

10. Laboratory for Computational Statistics Technical Report No. 7, *Smoothing with Split Linear Fits*, by J. A. McDonald and A. B. Owen, July 1984.

11. Mosteller, F. and Tukey, J.w., "Data Analysis Including Statistics", *Handbook of Social Psychology*, edited by G. Lindzey and E. Aronson, Addison-Wesley, Reading, Mass.

12. "Grafstat of the IBM 3277 Display," *Computer Center Newsletter*, v. 15, No. 4, pp. 13-17, May 18, 1983.

13. Spiegel, M. R., *Probability and Statistics*, McGraw-Hill, Inc., 1975.

14. Anscombe, F. J., *Computing in Statistical Science Through APL*, Springer-Verlag, 1981.

15. W. R. Church Computer Center, Naval Postgraduate School, *VS APL at NPS*, July 1982.

16. Chambers, J. M., *Computational Methods for Data Analysis*, J. Wiley and Sons, 1977.

BIBLIOGRAPHY

Boillot, M. and Shingles, C. R., *Understanding WATFIV*, West Publishing Co., 1980.

Clark, J. and Downing, D., *Statistics: The Easy Way*, Barron's Educational Series, Inc., 1983.

Fraser, D. C., *A New Technique For the Optimal Smoothing of Data*, Ph.D. Thesis, Massachusetts Institute of Technology, Boston, 1967.

Hancock, L. and Krieger, M., *The C Primer*, McGraw-Hill, 1982.

Herrero, J. L. and Willoner, G., *Synthesis of Filters*, Prentice-Hall, 1966.

Kelly, A. and Pohl, I., *An Introduction to Programming in C: A Book on C*, The Benjamin/Cummings Publishing Co., 1984.

Kendall, M. G. and Stuart, A., *The Advanced Theory of Statistics*, v. 2, Hafner, 1966.

Koopman, L. H., *The Spectral Analysis of Times Series*, Academic Press, 1974.

Morrison, N., *Introduction to Sequential Smoothing and Prediction*, McGraw-Hill, 1969.

Naval Postgraduate School, W. R. Church Computer Center Technical Memorandum, *SHERPA Laser Printer*, by Larry Frazier, August 1985.

Naval Postgraduate School, W. R. Church Computer Center Technical Note, TN-VM-04, *Using Exec Files Under CMS*, February 1984.

Naval Postgraduate School, W. R. Church Computer Center Technical Note, VM-01, *User's Guide to VM/CMS at NPS*, April 1985.

Naval Postgraduate School, W. R. Church Computer Center Technical Note, VM-06, *Script User's Guide For NPS*, September 1984.

Naval Postgraduate School, W. R. Church Computer Center Technical Note, VM-14, *Thesis9*, by Larry Frazier, November 1984.

Rhodes, E. C., *Tracts For Computers: No. VI-Smoothing*, edited by Pearson, K., Cambridge University Press, England, 1921.

Topping, J., *Errors of Observation and Their Treatment*, The Institute of Physics, London, 1956.

VS FORTRAN Programming Guide, SC 26-4118-0, 1st ed., published by IBM Corp., October 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Superintendent Attn: Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5100	2
3.	Dr. P. A. W. Lewis Department of Operations Research Code 55Lw Naval Postgraduate School Monterey, California 93943-5100	5
4.	Dr. Peter Welch IBM Research Laboratories Yorkstown Heights, N.Y. 10598	1
5.	Dr. Laurence C. Breaker Department of Oceanography Code 68By Naval Postgraduate School Monterey, California 93943-5100	1
6.	LCDR Paul S. Fischbeck Department of Operations Research Code 55Fb Naval Postgraduate School Monterey, California 93943-5100	1
7.	Art B. Owen Stanford Linear Accelerator Center Stanford, California 94512	1
8.	CPT. Jose A. Vasquez, Jr. HQ, USACECOM ATTN: AMSEL-PL-E Fort Monmouth, New Jersey 07703-5004	2